DEV-OPS INTERVIEW PREPERATION

1) Explain what is DevOps?

It is a newly emerging term in IT field, which is nothing but a practice that emphasizes the collaboration and communication of both software developers and other information-technology (IT) professionals. It focuses on delivering software product faster and lowering the failure rate of releases.

2) Mention what are the key aspects or principle behind DevOps?

The key aspects or principle behind DevOps is

- Infrastructure as code
- Continuous deployment
- Automation
- Monitoring
- Security

3) What are the core operations of DevOps with application development and with infrastructure?

The core operations of DevOps with

Application development

- Code building
- Code coverage
- Unit testing
- Packaging
- Deployment

With infrastructure

- Provisioning
- Configuration
- Orchestration
- Deployment

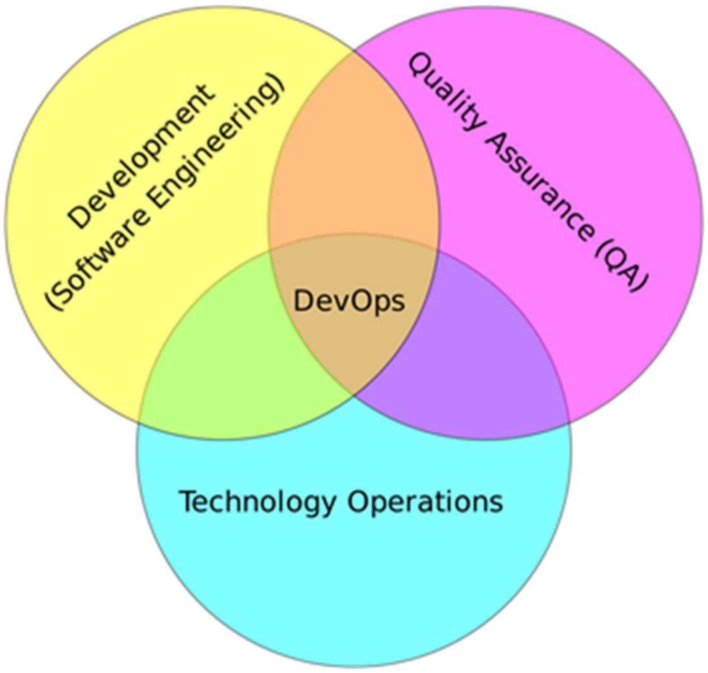
4) Explain how "Infrastructure of code" is processed or executed in AWS?

In AWS,

- The code for infrastructure will be in simple JSON format
- This JSON code will be organized into files called templates
- This templates can be deployed on AWS and then managed as stacks
- Later the CloudFormation service will do the Creating, deleting, updating, etc. operation in the stack

5) Explain which scripting language is most important for a DevOps engineer?

A simpler scripting language will be better for a DevOps engineer. Python seems to be very popular.



6) Explain how DevOps is helpful to developers?

DevOps can be helpful to developers to fix the bug and implement new features quickly. It also helps for clearer communication between the team members.

7) List out some popular tools for DevOps?

Some of the popular tools for DevOps are

- Jenkins
- Nagios
- Monit
- ELK (Elasticsearch, Logstash, Kibana)
- io
- Jenkins
- Docker
- Ansible
- Git
- Collectd/Collectl

8) Mention at what instance have you used the SSH?

I have used SSH to log into a remote machine and work on the command line. Beside this, I have also used it to tunnel into the system in order to facilitate secure encrypted communications between two untrusted hosts over an insecure network.

9) Explain how would you handle revision (version) control?

My approach to handle revision control would be to post the code on SourceForge or GitHub so everyone can view it. Also, I will post the checklist from the last revision to make sure that any unsolved issues are resolved.

10) Mention what are the types of Http requests?

The types of Http requests are

- GET
- HEAD
- PUT
- POST
- PATCH
- DELETE
- TRACE
- CONNECT
- OPTIONS

11) 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.

12) 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.

13. 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.

14. 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.

15. 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.

16. 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.

17. 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.

18. 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.

19. 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.

20. 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.

21. 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.

22. 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.

23. 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.

24. 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.

25. 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

26. 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.

27. 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.

28. 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.

29. 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.

30. 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.

31. 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.

32. 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.

33. 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.

34. 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.

35. 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.

36. 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.

37. 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 intervewer that you are agile according to the needs of your organization.

38. What problems did you face and how did you solve them in a way that met the team's goals?
Tips to answer: This questions 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.

39. 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.

40. 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

41) Explain what is DevOps?

It is a newly emerging term in IT field, which is nothing but a practice that emphasizes the collaboration and communication of both software developers and other information-technology (IT) professionals. It focuses on delivering software product faster and lowering the failure rate of releases.

42) Mention what are the key aspects or principle behind DevOps?

The key aspects or principle behind DevOps is

- Infrastructure as code

- Continuous deployment

- Automation

- Monitoring

- Security

43) What are the core operations of DevOps with application development and with infrastructure?

The core operations of DevOps with

Application development

- Code building

- Code coverage

- Unit testing

- Packaging

- Deployment

With infrastructure

- Provisioning

- Configuration

- Orchestration

- Deployment

44) Explain how "Infrastructure of code" is processed or executed in AWS?

In AWS,

- The code for infrastructure will be in simple JSON format

- This JSON code will be organized into files called templates

- This templates can be deployed on AWS and then managed as stacks

- Later the CloudFormation service will do the Creating, deleting, updating, etc. operation in the stack

45) Explain which scripting language is most important for a DevOps engineer?

A simpler scripting language will be better for a DevOps engineer. Python seems to be very popular.

46) Explain how DevOps is helpful to developers?

DevOps can be helpful to developers to fix the bug and implement new features quickly. It also helps for clearer communication between the team members.

47) List out some popular tools for DevOps?

Some of the popular tools for DevOps are

- Jenkins

- Nagios

- Monit

- ELK (Elasticsearch, Logstash, Kibana)

- io

- Jenkins

- Docker

- Ansible

- Git

- Collectd/Collectl

48) Mention at what instance have you used the SSH?

I have used SSH to log into a remote machine and work on the command line. Beside this, I have also used it to tunnel into the system in order to facilitate secure encrypted communications between two untrusted hosts over an insecure network.

49) Explain how would you handle revision (version) control?

My approach to handle revision control would be to post the code on SourceForge or GitHub so everyone can view it. Also, I will post the checklist from the last revision to make sure that any unsolved issues are resolved.

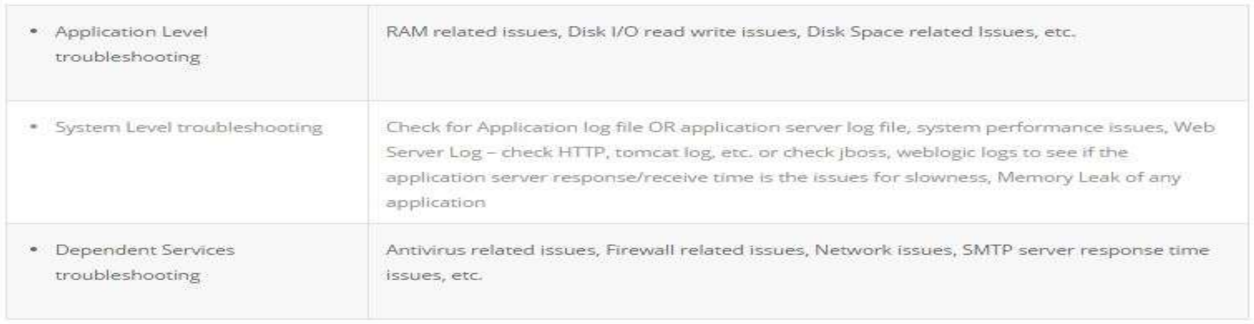
50) Mention what are the types of Http requests?

The types of Http requests are

- GET	- HEAD	- PUT
- POST	- PATCH	- DELETE
- TRACE	- CONNECT	- OPTIONS

51) Explain what would you check If a Linux-build-server suddenly starts getting slow?

If a Linux-build-server suddenly starts getting slow, you will check for following three things



• Application Level troubleshooting	RAM related issues, Disk I/O read write issues, Disk Space related Issues, etc.
• System Level troubleshooting	Check for Application log file OR application server log file, system performance issues, Web Server Log – check HTTP, tomcat log, etc. or check jboss, weblogic logs to see if the application server response/receive time is the issues for slowness, Memory Leak of any application
• Dependent Services troubleshooting	Antivirus related issues, Firewall related issues, Network issues, SMTP server response time issues, etc.

52) How does HTTP work? How does a web page appear in a browser?

Basically, to prove that you understand (at a high-level) the infrastructure you'll be working with, as well as the ability to explain concepts to others. DevOps is about communication too, remember?

53) What are some examples of how you might scale a write/read-heavy application? Why?

Do you know how to scale different kinds of applications? How would you identify bottlenecks? Do you have a reasonable understanding of how to scale an entire environment, or migrate to the cloud (or another cloud) if necessary?

54) Tell me about the worst-run/best-run outage you've been a part of. What made it bad/well-run?

Take lessons from both failures and successes in dealing with outages. Remember, there's no shame in talking about the mistakes you've learned from.

55) How would you assess how "deployable" a system is?

How do you know when a deployment is ready? How would you manage that deployment?

56) How would you prepare for a migration from one platform to another?

Fairly self-explanatory, how do you handle transitions? How do you document the current state of your environment before moving, and how do you validate everything is as it should be after the fact?

57) What is the purpose of a post-mortem meeting?

Do you know how to learn from mistakes? How do you get the most value out of post-mortems?

58) How would you make key aspects of a software system traceable?

So that you can figure out what happened quickly if/when something goes wrong. What types of monitoring and alerting are most important to you? What tooling do you use?

59) How do you handle interruptions?

While it's going to be your job to *stop* interruptions, how you deal with them now is important for getting an idea of if you know how to fix them quickly and if you know how to make devs and ops work together.

60) How would you deploy software to 5000 systems?

Do you know how you would go about large-scale deployments?

61) What different types of testing need to be carried out on a software system, and what tools would you use to achieve this testing?

For starters, you probably would want to test that your configurations are correct before and after making changes, and know how to validate that your environments are all consistent.

62) 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.

63) 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.

64) 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.

65) 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.

66) 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.

67) 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.

68) 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.

69) What other tools might help you in this role?

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.

70) 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.

71. How does HTTP work? How does a web page appear in a browser?

Basically, to prove that you understand (at a high-level) the infrastructure you'll be working with, as well as the ability to explain concepts to others. DevOps is about communication too, remember?

72. What are some examples of how you might scale a write/read-heavy application? Why?

Do you know how to scale different kinds of applications? How would you identify bottlenecks? Do you have a reasonable understanding of how to scale an entire environment, or migrate to the cloud (or another cloud) if necessary?

73. Tell me about the worst-run/best-run outage you've been a part of. What made it bad/well-run?

Take lessons from both failures and successes in dealing with outages. Remember, there's no shame in talking about the mistakes you've learned from.

74. How would you assess how "deployable" a system is?

How do you know when a deployment is ready? How would you manage that deployment?

75. How would you prepare for a migration from one platform to another?

Fairly self-explanatory, how do you handle transitions? How do you document the current state of your environment before moving, and how do you validate everything is as it should be after the fact?

76. What is the purpose of a post-mortem meeting?

Do you know how to learn from mistakes? How do you get the most value out of post-mortems?

77. How would you make key aspects of a software system traceable?

So that you can figure out what happened quickly if/when something goes wrong. What types of monitoring and alerting are most important to you? What tooling do you use?

78. How do you handle interruptions?

While it's going to be your job to *stop* interruptions, how you deal with them now is important for getting an idea of if you know how to fix them quickly and if you know how to make devs and ops work together.

79. How would you deploy software to 5000 systems?

Do you know how you would go about large-scale deployments?

80. What different types of testing need to be carried out on a software system, and what tools would you use to achieve this testing?

For starters, you probably would want to test that your configurations are correct before and after making changes, and know how to validate that your environments are all consistent. Maybe you should try UpGuard? It's free to get started. ;)

81) Explain what would you check If a Linux-build-server suddenly starts getting slow?

If a Linux-build-server suddenly starts getting slow, you will check for following three things

• Application Level troubleshooting	RAM related issues, Disk I/O read write issues, Disk Space related Issues, etc.
• System Level troubleshooting	Check for Application log file OR application server log file, system performance issues, Web Server Log – check HTTP, tomcat log, etc. or check jboss, weblogic logs to see if the application server response/receive time is the issues for slowness, Memory Leak of any application
• Dependent Services troubleshooting	Antivirus related issues, Firewall related issues, Network issues, SMTP server response time issues, etc.

82. How does HTTP work? How does a web page appear in a browser?

Basically, to prove that you understand (at a high-level) the infrastructure you'll be working with, as well as the ability to explain concepts to others. DevOps is about communication too, remember?

83. What are some examples of how you might scale a write/read-heavy application? Why?

Do you know how to scale different kinds of applications? How would you identify bottlenecks? Do you have a reasonable understanding of how to scale an entire environment, or migrate to the cloud (or another cloud) if necessary?

84. Tell me about the worst-run/best-run outage you've been a part of. What made it bad/well-run?

Take lessons from both failures and successes in dealing with outages. Remember, there's no shame in talking about the mistakes you've learned from.

85. How would you assess how "deployable" a system is?

How do you know when a deployment is ready? How would you manage that deployment?

86. How would you prepare for a migration from one platform to another?

Fairly self-explanatory, how do you handle transitions? How do you document the current state of your environment before moving, and how do you validate everything is as it should be after the fact?

87. What is the purpose of a post-mortem meeting?

Do you know how to learn from mistakes? How do you get the most value out of post-mortems?

88. How would you make key aspects of a software system traceable?

So that you can figure out what happened quickly if/when something goes wrong. What types of monitoring and alerting are most important to you? What tooling do you use?

88. How do you handle interruptions?

While it's going to be your job to *stop* interruptions, how you deal with them now is important for getting an idea of if you know how to fix them quickly and if you know how to make devs and ops work together.

89. How would you deploy software to 5000 systems?

Do you know how you would go about large-scale deployments?

90. What different types of testing need to be carried out on a software system, and what tools would you use to achieve this testing?

For starters, you probably would want to test that your configurations are correct before and after making changes, and know how to validate that your environments are all consistent. Maybe you should try UpGuard? It's free to get started. ;)

91. How does HTTP work…?

This *ought* to be just another of the 20-30 technical questions which every second-interview candidate should be able to answer with ease, but *very few candidates can answer this question*; those than can are among those that "get it". The answers I look for include a discussion of the OSI model, TCP connections (preferably with a mention of Keep-Alive), web server technology, MIME types, and so on.
In my experience, any candidate who fails at this hurdle is likely to be too "lightweight" to be of great value, because good answers require a "**top to bottom" understanding of what is really happening with web-based software**.

92. How would you prepare for migration…?

This question evaluates the candidate's experience of real projects with all the awkwardness and complexity they bring. Developing greenfield systems with little or no existing technology in place is always easier than having to deal with legacy components and configuration, and candidates who appreciate that any interesting software system will in effect be under **constant migration** are well-suited for DevOps roles.
I like to hear discussion of cut-over, dress rehearsals, roll-back and roll-forward, DNS solutions, feature toggles, branch by abstraction, and automation.

93. What types of testing are needed…?

Too often, members of software teams (usually developers) will 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 like to find team members who practise defensive programming in a pragmatic way, which often means assuming that the code will fail and planning for those failures.
This question also exposes the level of experience the candidate has, and their**exposure to real-world failure modes**. I look for mentions of: unit test strategy; use of test harnesses; early load testing; network simulation; A/B and multi-variate testing; etc.

94. How would you ensure traceability…?

The ability to trace or track behaviour of a software system is sometimes bolted-on as an after-thought. This question probes the candidate's attitude to metrics, logging, transaction journeys, and reporting; the ideal candidate will identify that**metrics, monitoring and logging need to be a core part of the software system**, and that without them, the software is essentially unmaintainable and undiagnosable.
The words SysLog, Splunk, error tracking, Nagios, SCOM, Avicode and so on are what I'm looking for here.

95. How would you make software deployable…?

The ability to script the installation and reconfiguration of software systems is essential to controlled, automated change. Although (rightly) 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.

Software team members who appreciate the need to **expose configuration and settings in a manner accessible to automation** will talk about Inversion of Control (IoC) and Dependency Injection, scripted installation, test harnesses, separation of concerns, command-line tools, and "infrastructure as code".

Docker Interview Questions

What is Docker
Docker is a platform to run each application isolated and securely. Internally it achieves it by using kernel containerization feature.

What is the advantage of Docker over hypervisors?
Docker is light weight and more efficient in terms of resource uses because it uses the host underlying kernel rather than creating its own hypervisor.

What is Docker Image
Docker Image is the source of the docker container. In other words, docker images are used to create containers. It is possible create multiple isolated containers from a single image.

What is Docker Container?
Docker Container is the instantiation of docker image. In other words, it is the run time instance of images. Images are set of files whereas containers is the one who run the image inside isolated.

Difference between Docker Image and container?
Docker container is the runtime instance of docker image. Docker Image does not have a state and its state never changes as it is just set of files whereas docker container has its execution state.

What DevOps tools have you worked with?

DevOps involves a constant cycle of coding, testing, deployment and refinements. The right tools depend on team size and specializations within the organization.

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 principal languages -- 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 done with cloud-based software development?

Expect the next DevOps interview question to cover your experience with cloud services, which dovetail with DevOps' iterative, integrated IT approach.

Many DevOps professionals need hands-on experience with major public cloud providers like Amazon Web Services (AWS) Elastic Compute Cloud and Google Cloud Platform. 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.

A comprehensive understanding of third-party tools underscores your ability to bring DevOps to cloud projects. Focus on your successful projects or tasks using cloud rather than just listing off products.

In your interview, mention experience handling big data, which uses highly scalable cloud infrastructures to tackle complex computing tasks.

Give me an example of how you handle projects?

Underscore your exposure to DevOps project management issues in this answer. Frame the discussion of DevOps management around solving complex business problems.

As a professional with managerial responsibilities, you must demonstrate a clear understanding of DevOps project management tactics. Discuss how you work with teams to set objectives, streamline workflow, maintain scope (prevent creep), research and introduce new tools or frameworks, translate requirements into workflow and follow up. Include how CI, and release management and other tools keep interdisciplinary projects on track.

High-level DevOps roles, such as chief DevOps engineer, require more business acumen and people skills. Share experiences you've had resolving team disputes and conflicts.

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

This DevOps interview question can really elevate you from a "qualified" to an "ideal candidate."

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.

Organizations that promote a standardized approach to software development seek candidates familiar with Agile application lifecycle management techniques and tools.

And DevOps roles listing a strong operations focus in a regulated industry may go to the candidate familiar with ITIL practices.

1) 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.

- **What Most People Say:** "The DevOps role entailed working in a silo, doing just infrastructure automation or release management by pushing development and the ops team."
- **What You Should Say:** "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."
- **Why You Should Say It:** Display a flexible mindset. Know the latest and greatest software engineering techniques and explain how you'll use those tools and skills to solve problems the interviewer throws your way. It's also important to express a passion and curiosity to learn something new and a willingness to tackle new challenges. It's all about a willingness to adapt.

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

- **What Most People Say:** "I've found that people in IT Ops, QA and development are constantly at each other's throats. That's why organizations need people like me, to diagnose the problem and prevent projects from stagnating."
- **What You Should Say:** "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."
- **Why You Should Say It:** It's smart to avoid sounding too judgmental or inflexible. Make sure your point of view isn't too deeply rooted in one side or the other—development or operations. You don't want to give the impression that you're a lone wolf who can't work collaboratively, and who seems to want to take too much of the credit. Show that you can think objectively while not pointing fingers. Show eagerness to employ all available collaboration techniques to effectively communicate between teams.

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

- **What Most People Say:** “I have a desire to move into development. That’s my eventual career goal.”
- **What You Should Say:** “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.”
- **Why You Should Say It:** No employer wants to hire someone whose only goal is to move quickly into another role, or who lacks energy and enthusiasm for the job under discussion. Your passion and excitement will show your potential employer that you’re a great fit for DevOps. These are important soft skills that are at the heart of the DevOps engineer, and they lay the groundwork for future moves.

4) What testing is necessary to insure a new service is ready for production?

- **What Most People Say:** “I think testing is really a responsibility of the QA department. Developers really need to be focused on writing code.”
- **What You Should Say:** “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.”
- **Why You Should Say It:** It’s important to communicate to your interviewer that you understand how DevOps’ approach benefits the entire process holistically, and that you don’t think in terms of conventional silos.

chef configuration management interview questions and answers

1.What is a resource?
Answer- A resource represents a piece of infrastructure and its desired state, such as a package that should be installed, a service that should be running, or a file that should be generated.

2.Question: What is a recipe?
Answer- A recipe is a collection of resources that describes a particular configuration or policy. A recipe describes everything that is required to configure part of a system. Recipes do things such as:

install and configure software components.
manage files.
deploy applications.
execute other recipes.

3.Question: What happens when you don't specify a resource's action?
Answer- When you don't specify a resource's action, Chef applies the default action.

4.Question: Are these two recipes the same?

```
package 'httpd'
service 'httpd' do
   action [:enable, :start]
 end
```

&&

```
service 'httpd' do
   action [:enable, :start]
   end
package 'httpd'
```

Answer-
No, they are not. Remember that Chef applies resources in the order they appear. So the first recipe ensures that the httpd package is installed and then configures the service. The second recipe configures the service and then ensures the package is installed.

5.Question: The second recipe may not work as you'd expect because the service resource will fail if the package is not yet installed.

Are these two recipes the same?

```
package 'httpd'
service 'httpd' do
   action [:enable, :start]
   end
package 'httpd'
service 'httpd' do
   action [:start, :enable]
   end
```

Answer-
No, they are not. Although both recipes ensure that the httpd package is installed before configuring its service, the first recipe enables the service when the system boots and then starts it. The second recipe starts the service and then enables it to start on reboot.

Are these two recipes the same?

```
file '/etc/motd' do
owner 'root'
group 'root'
mode '0755'
action :create
end
```

```
file '/etc/motd' do
action :create
mode '0755'
group 'root'
owner 'root'
end
```

Answer-
Yes, they are! Order matters with a lot of things in Chef, but you can order resource attributes any way you want.

6.Question - Write a service resource that stops and then disables the httpd service from starting when the system boots.

Answer -
```
service 'httpd' do
action [:stop, :disable]
end
```

7.How does chef-apply differ from chef-client?

chef-apply applies a single recipe; chef-client applies a cookbook.

For learning purposes, we had you start off with chef-apply because it helps you understand the basics quickly. In practice, chef-apply is useful when you want to quickly test something out. But for production purposes, you typically run chef-client to apply one or more cookbooks.

You'll learn in the next module how to run chef-client remotely from your workstation.

8.How does a cookbook differ from a recipe?
A recipe is a collection of resources, and typically configures a software package or some piece of infrastructure. A cookbook groups together recipes and other information in a way that is more manageable than having just recipes alone.

For example, in this lesson you used a template resource to manage your HTML home page from an external file. The recipe stated the configuration policy for your web site, and the template file contained the data. You used a cookbook to package both parts up into a single unit that you can later deploy.

9.What's the run-list?

The run-list lets you specify which recipes to run, and the order in which to run them. The run-list is important for when you have multiple cookbooks, and the order in which they run matters.

10.What are the two ways to set up a Chef server?

Install an instance on your own infrastructure.
Use hosted Chef.

11.What's the role of the Starter Kit?
The Starter Kit provides certificates and other files that enable you to securely communicate with the Chef server.

12.Where can you get reusable cookbooks that are written and maintained by the Chef community?
Chef Supermarket, https://supermarket.chef.io.

13.What's the command that enables you to interact with the Chef server?
knife

14.What is a node?
A node represents a server and is typically a virtual machine,

container instance, or physical server – basically any compute resource in your infrastructure that's managed by Chef.

15.What information do you need to in order to bootstrap?
You need:

your node's host name or public IP address.
a user name and password you can log on to your node with.
Alternatively, you can use key-based authentication instead of providing a user name and password.

16.What happens during the bootstrap process?
During the bootstrap process, the node downloads and installs chef-client, registers itself with the Chef server, and does an initial checkin. During this checkin, the node applies any cookbooks that are part of its run-list.

17.Which of the following lets you verify that your node has successfully bootstrapped?

The Chef management console.
knife node list
knife node show
You can use all three of these methods.

18.What is the command you use to upload a cookbook to the Chef server?
knife cookbook upload

19.How do you apply an updated cookbook to your node?
We mentioned two ways.

Run knife ssh from your workstation.
SSH directly into your server and run chef-client.
You can also run chef-client as a daemon, or service, to check in with the Chef server on a regular interval, say every 15 or 30 minutes.

Update your Apache cookbook to display your node's host name, platform, total installed memory, and number of CPUs in addition to its FQDN on the home page.

Update index.html.erb like this.
```
<html>
```

```
<body>
<h1>hello from <%= node['fqdn'] %></h1>

<pre>
<%= node['hostname'] %>
<%= node['platform'] %> - <%= node['platform_version'] %>
<%= node['memory']['total'] %> RAM
<%= node['cpu']['total'] %> CPUs
</pre>
</body>
</html>
```

Then upload your cookbook and run it on your node.

20.What would you set your cookbook's version to once it's ready to use in production?

According to Semantic Versioning, you should set your cookbook's version number to 1.0.0 at the point it's ready to use in production.

21.What is the latest version of the haproxy community cookbook?

To know the latest version of any cookbook on Chef Supermarket, browse to its page and view the latest version from the version selection box.

Or, get the info from the knife cookbook site command, like this.

```
knife cookbook site show haproxy | grep latest_version
latest_version:
http://cookbooks.opscode.com/api/v1/cookbooks/haproxy/versions/1
.6.6
```

22.Create a second node and apply the awesome_customers cookbook to it. How long does it take?

You already accomplished the majority of the tasks that you need. You wrote the awesome_customers cookbook, uploaded it and its dependent cookbooks to the Chef server, applied the awesome_customers cookbook to your node, and verified that everything's working.

All you need to do now is:

Bring up a second Red Hat Enterprise Linux or CentOS node.
Copy your secret key file to your second node.
Bootstrap your node the same way as before. Because you include the awesome_customers cookbook in your run-list, your node will apply that cookbook during the bootstrap process.
The result is a second node that's configured identically to the first one. The process should take far less time because you already did most of the work.

Now when you fix an issue or add a new feature, you'll be able to deploy and verify your update much more quickly!

23.What's the value of local development using Test Kitchen?

Local development with Test Kitchen:

enables you to use a variety of virtualization providers that create virtual machine or container instances locally on your workstation or in the cloud.
enables you to run your cookbooks on servers that resemble those that you use in production.
speeds up the development cycle by automatically provisioning and tearing down temporary instances, resolving cookbook dependencies, and applying your cookbooks to your instances.

24.What is VirtualBox? What is Vagrant?

VirtualBox is the software that manages your virtual machine instances.

Vagrant helps Test Kitchen communicate with VirtualBox and configures things like available memory and network settings.

Verify that your motd cookbook runs on both CentOS 6.6 and CentOS 6.5.

Your motd cookbook is already configured to work on CentOS 6.6 as well as CentOS 6.5, so you don't need to modify it.

To run it on CentOS 6.5, add an entry to the platforms section of your .kitchen.yml file like this.

```
---
      driver:
      name: vagrant
provisioner:
      name: chef_zero
platforms:
      - name: centos-6.6
      driver:
      box: opscode-centos-6.6
      box_url: http://opscode-vm-
bento.s3.amazonaws.com/vagrant/virtualbox/opscode_centos-
6.6_chef-provisionerless.box
      - name: centos-6.5
      driver:
      box: opscode-centos-6.5
      box_url: http://opscode-vm-
bento.s3.amazonaws.com/vagrant/virtualbox/opscode_centos-
6.5_chef-provisionerless.box
suites:
      - name: default
      run_list:
      - recipe[motd::default]
      attributes:
```

In many cases, Test Kitchen can infer the box and box_url parameters, which specify the name and location of the base image, or box. We specify them here to show you how to use them.

Run kitchen list to see the matrix of test instances that are available. Here, we have two platforms – CentOS 6.5 and CentOS 6.6 – multiplied by one suite – default.

```
$kitchen list


Instance           Driver   Provisioner  Verifier  Transport
Last Action
default-centos-66  Vagrant  ChefZero     Busser    Ssh
<Not Created>
default-centos-65  Vagrant  ChefZero     Busser    Ssh
<Not Created>
```

Run kitchen converge to create the instances and apply the motd cookbook.

```
$kitchen converge
   -----> Starting Kitchen (v1.4.0)
   -----> Creating <default-centos-66>...
   Bringing machine 'default' up with 'virtualbox' provider...
   [...]
   Running handlers:
   Running handlers complete
   Chef Client finished, 1/1 resources updated in 10.372334751
seconds
   Finished converging <default-centos-66> (3m52.59s).
   -----> Creating <default-centos-65>...
   Bringing machine 'default' up with 'virtualbox' provider...
   [...]
   Running handlers:
   Running handlers complete
   Chef Client finished, 1/1 resources updated in 5.32753132
seconds
   Finished converging <default-centos-65> (10m12.63s).
 -----> Kitchen is finished. (19m47.71s)
```

Now to confirm that everything's working, run kitchen login. But this time, you need to provide the instance name so that Test Kitchen knows which instance to connect to.

```
$kitchen login default-centos-66
      Last login: Wed May 13 20:15:00 2015 from 10.0.2.2

      hostname:  default-centos-66
      fqdn:      default-centos-66
      memory:    469392kB
      cpu count: 1
      [vagrant@default-centos-66 ~]$ logout
      Connection to 127.0.0.1 closed.$kitchen login default-
centos-65
      Last login: Wed May 13 20:28:18 2015 from 10.0.2.2

      hostname:  default-centos-65
      fqdn:      default-centos-65
      memory:    469452kB
      cpu count: 1
      [vagrant@default-centos-65 ~]$ logout
      Connection to 127.0.0.1 closed.
```

If a linux-build-server suddenly starts getting slow, what would you check?

If a linux-build-server suddenly starts getting slow, I would divide my approach / troubleshooting into 3 section as follows;

1. System Level troubleshooting

a. RAM related issues
b. Disk Space related Issues
c. Disk I/O read write issues
d. Network Hardware issues
e. Mount issues
f. Too Many process running in the machine

2. Application Level troubleshooting

a. Application is not behaving properly. Hit to Application log file OR application server log file OR web server Log file and try to understand the issues.
b. zombie process issues - Find out if any as such process which is causing the system performance issues.
c. Application Log - depends on the application installed, this can be referred and make use of the experience with the project and troubleshoot.
d. Web Server Log - we can check http, tomcat log as well.
e. Application Server Log - We can see jboss, weblogic logs to see if the application server response/receive time is the issues for slowness.

f. Memory Leak of any application - This is one of well known issues in lunux based server due to bad application coding. Many times this can be resolved either by fixing the code or rebooting. But many other solutions are there to apply.

3. Dependent Services troubleshooting

a. SMTP Response time - SMTP server is not responding faster which is causing delay in response and queue up many processes.
b. Network issues - There are many System performance issues is dependent on network or service which is depends on the network.
c. Firewall related issues

d. Antivirus related issues

Some of the useful commands for troubleshooting are..

1. df -k
2. du -sh
3. top
4. uptime
5. ps -eaf | grep
6. vmstat
7. ping
8. tail -f <logfile>
9. iostat
10. free
11. kill -9
12. mount
13. sar
14. ifconfig eth0 | enable | disable
15. traceroute
16. netstat -r
17. nslookup
18. route

There are various options available for each commands and depends on the need during the troubleshooting, right commands can be applied. To find arguments and their explanation, I usually refer google.com rather than man page.

Interview Questions and Answer for Perforce Version Control Tool

1.Some of the perforce commands which is not commonly used but useful.
p4 annotate - Print file lines along with their revisions.
e.g p4 annotate file.c

2.How to ignore files/folder in perforce workspace/client?
Assuming you have a client named "CLIENT", a directory named "foo" (located at your project root), and you wish to ignore all .dll files in that directory tree, you can add the following lines to your workspace view to accomplish this:

```
-//depot/foo/*.dll //CLIENT/foo/*.dll
-//depot/foo/.../*.dll //CLIENT/foo/.../*.dll
```

The first line removes them from the directory "foo" and the second line removes them from all sub directories. Now, when you 'Reconcile Offline Work...', all the .dll files will be moved into "Excluded Files" folders at the bottom of the folder diff display. They will be out of your way, but can still view and manipulate them if you really need to.

You can also do it another way, which will reduce your "Excluded Files" folder to just one, but you won't be able to manipulate any of the files it contains because the path will be corrupt (but if you just want them out of your way, it doesn't matter).

```
-//depot/foo.../*.dll //CLIENT/foo.../*.dll
```

Reference
How can I exclude a directory from a Perforce command?

P4IGNORE
Specify a file that contains a list of files to ignore when adding files to the depot and reconciling workspaces.
Reference

3.How to check last 10 submitted, pending, or shelved changelists that include any file under the project directory?
```
p4 changes -m 5 //depot/project/...
```

4.How to check last 10 submitted or pending, or shelved changelists that include any file under the project directory?
```
p4 changes -m 1 -s submitted | pending | shelved
```

5.Interview Questions Related to Perforce Admin

1. How to take perforce backup
2. How to restore perforce backup
3. How to verify health of the perforce repos database
4. What is the ise of p4 dbverify and p4 verify

6.What is the use of p4 admin commands.

The p4 admin command allows Perforce superusers to perform administrative tasks even when working from a different machine than the one running the shared Perforce service.
```
p4 [g-opts] admin checkpoint [ -z | -Z ] [ prefix ]
p4 [g-opts] admin journal [ -z ] [ prefix ]
p4 [g-opts] admin stop
```

```
p4 [g-opts] admin restart
p4 [g-opts] admin updatespecdepot [ -a | -s type ]
p4 [g-opts] admin resetpassword -a | -u user
```

Reference – [Click here]()

7.How to remove files from perforce permanently?

p4 archive -p with caution. This is the one of only two commands in Perforce that actually removes file data. (The other command that removes file data is p4 obliterate.)

8.How to set properly in Perforce?

The Perforce service offers three ways of storing metadata: counters/keys, attributes, and properties.

If your application requires only the flat storage of simple key/value pairs, and attempts to implement no security model, use the p4 counters and p4 keys commands.

The p4 property command can be used by administrators to view and update property definitions stored in the Perforce service. The service does not use the property definitions; it provides this capability for other Perforce applications, such as P4V

If your application's metadata is associated with particular files, use p4 attribute.

If your application's metadata is not associated with files, and if you have a requirement to restrict its visibility to users, groups, and/or to control the precedence of multiple values using sequence numbers, use p4 property.

```
p4 property -n name -v value
p4 counter mycounter 123
p4 key mykey 12
p4 attribute -n name [ -v value ] files...
```

9.Perforce Integration with other Tools

1. Gitfushion
2. Swarm
3. Replication

Puppet Interview Questions & Answers

Describe the most significant gain you made from automating a process through Puppet.

- **What Most People Say:** "I automated the configuration and deployment of Linux and Windows machines using Puppet. The process used to take a week and now it happens in less than 10 minutes."
- **What You Should Say:** "I automated the configuration and deployment of Linux and Windows machines using Puppet. In addition to shortening the processing time from one week to 10 minutes, I used the roles and profiles paradigm and documented the purpose of each module in README to ensure that others could update the module using Git. The modules I wrote are still being used, but they've been improved by my teammates and members of the community."
- **Why You Should Say It:** Automation is not just about the initial efficiency gain—it's about long-term success. Therefore, it's critical to demonstrate sustained improvements by minimizing and simplifying manual intervention not just for routine or daily changes, but for major events such as upgrades, process changes and personnel changes.

Tell me about a time when you used collaboration and Puppet to help resolve a conflict within a team.

- **What Most People Say:** "The development team wanted root access on test machines managed by Puppet in order to make configuration changes. We didn't want to give them root access because we wanted to manage consistency across testing boxes. I worked with the development team and figured out a way for developers to change only specific configuration values that would feed into the machines they used."
- **What You Should Say:** "The development team wanted root access on test machines managed by Puppet in order to make specific configuration changes. We responded by meeting with them weekly to agree on a process for developers to communicate

configuration changes and to empower them to make many of the changes they needed. Through our joint efforts, we came up with a way for the developers to change specific configuration values themselves via data abstracted through Hiera. In fact, we even taught one of the developers how to write Puppet code in collaboration with us.”

- **Why You Should Say It:** Using Puppet robustly to serve other teams throughout your organization means breaking down organizational silos. While Puppet centralizes control and management of infrastructure, the modular nature of the DSL means that it should be reusable, extensible and serve as a platform for cross-team collaboration.

Which open source or community tools do you use to make Puppet more powerful?

- **What Most People Say:** “I use Git and Puppet’s Code Manager app to manage Puppet code in accordance with best practices.”
- **What You Should Say:** “Changes and requests are ticketed through Jira and we manage requests through an internal process. Then, we use Git and Puppet’s Code Manager app to manage Puppet code in accordance with best practices. Additionally, we run all of our Puppet changes through our continuous integration pipeline in Jenkins using the beaker testing framework.”
- **Why You Should Say It:** A strong Puppet practitioner will definitely utilize several community tools as well as best practices to enhance Puppet’s performance. He or she will also use version control, code review and change management through a CI pipeline with robust testing.

The real masters are resourceful when it comes to working with technology and others. It’s important to demonstrate your mastery of Puppet automation, teamwork, collaboration and reusable tooling when you interview for any DevOps position.

1. What is Puppet ?

Puppet is a configuration Tool which is use to automate administration tasks.Puppet Agent(Client) sends request to Puppet Master (Server) and Puppet Master Push Configuration on Agent.

2. What is Manifests ?

Manifests, in Puppet, are the files in which the client configuration is specified.

3. What is Module and How it is different from Manifest ?

Whatever the manifests we defined in modules, can call or include into other manifests. Which makes easier management of Manifests.It helps you to push specific manifests on specific Node or Agent.

4. Command to check requests of Certificates ?

```
puppetca –list (2.6)
puppet ca list (3.0)
```

5. Command to sign Requested Certificates

```
puppetca –sign hostname-of-agent (2.6)
puppet ca  sign hostname-of-agent (3.0)
```

6. Where Puppet Master Stores Certificates

```
/var/lib/puppet/ssl/ca/signed
```

7. What is Facter ?

Sometime you need to write manifests on conditional experession based on agent specific data which is available through Facter. Facter provides information like Kernel version,Dist release, IP Address, CPU info and etc.You can defined your facter also.

8. What is the use of etckeeper-commit-post and etckeeper-commit-pre on Puppet Agent ?

etckeeper-commit-post: In this configuration file you can define command and scripts which executes after pushing configuration on Agent
Etckeeper-commit-pre: In this configuration file you can define command and scripts which executes before pushing configuration on Agent.

9. What is Puppet Kick ?

By default Puppet Agent request to Puppet Master after a periodic time which known as “runinterval”. Puppet Kick is a utility which allows you to trigger Puppet Agent from Puppet Master.

10. What is MCollective ?

MCollective is a powerful orchestration framework. Run actions on thousands of servers simultaneously, using existing plugins or writing your own.

11. What’s special about Puppet’s model-driven design?

Traditionally, managing the configurations of a large group of computers has meant a series of imperative steps; in its rawest state, SSH and a *for* loop. This general approach grew more sophisticated over time, but it retained the more profound limitations at its root.

Puppet takes a different approach, which is to model everything — the current state of the node, the desired configuration state, the actions taken during configuration enforcement — as data: each node receives a catalog of resources and relationships, compares it to the current system state, and makes changes as needed to bring the system into compliance.

The benefits go far beyond just healing the headaches of configuration drift and unknown system state: modeling systems as data lets Puppet simulate configuration changes, track the history of a system over its lifecycle, and prove that refactored manifest code still produces the same system state. It also drastically lowers the barrier to entry for hacking and extending Puppet: instead of analyzing code and reverse-engineering the effects of each step, a user can just parse data, and sysadmins have been able to add significant value to their Puppet deployments with an afternoon’s worth of perl scripting.

12. Why does Puppet have its own language?

Why not use XML or YAML as the configuration format? Why not use Ruby as the input language?

The language used for manifests is ultimately Puppet’s human interface, and XML and YAML, being data formats developed around the processing capabilities of computers, are horrible human interfaces. While some people are comfortable reading and writing them, there’s a reason why we use web browsers instead of just reading the HTML directly. Also, using XML or YAML would limit any assurance that the interface was declarative — one process might treat an XML configuration differently from another.

13. Can Puppet manage workstations?

Yes: Puppet can manage any type of machine, and is used to manage many organizations that have a mix of laptops and desktops.

14. Does Puppet run on Windows?

Yes. As of Puppet 2.7.6 basic types and providers do run on Windows, and the test suite is being run on Windows to ensure future compatibility. More information can be found on the Puppet on Windows page, and bug reports and patches are welcome.

15. What size organizations should use Puppet?

There is no minimum or maximum organization size that can benefit from Puppet, but there are sizes that are more likely to benefit. Organizations with only a handful of servers are unlikely to consider maintaining those servers to be a real problem, while those that have more need to consider carefully how they eliminate manual management tasks.

16. My servers are all unique; can Puppet still help?

Yes.

All servers are at least somewhat unique, but very few servers are entirely unique; host names and IP addresses (e.g.) will always differ, but nearly every server runs a relatively standard operating system. Servers are also often very similar to other servers within a single organization — all Solaris servers might have similar security settings, or all web servers might have roughly equivalent configurations — even if they're very different from servers in other organizations. Finally, servers are often needlessly unique, in that they have been built and managed manually with no attempt at retaining appropriate consistency.

Puppet can help both on the side of consistency and uniqueness. Puppet can be used to express the consistency that should exist, even if that consistency spans arbitrary sets of servers based on any type of data like operating system, data centre, or physical location. Puppet can also be used to handle uniqueness, either by allowing special provision of what makes a given host unique or through specifying exceptions to otherwise standard classes.

17. Who is Puppet Labs?

Puppet Labs (formerly Reductive Labs) is a small, private company focused on re-framing the server automation problem.

18. How should I upgrade Puppet and Facter?

The best way to install and upgrade Puppet and Facter is via your operating system’s package management system, using either your vendor’s repository or one of Puppet Labs’ public repositories.

If you have installed Puppet from source, make sure you remove old versions entirely (including all application and library files) before upgrading. Configuration data (usually located in/etc/puppet or /var/lib/puppet, although the location can vary) can be left in place between installs.

19. What characters are permitted in a class name? In a module name? In other identifiers?

Class names can contain lowercase letters, numbers, and underscores, and should begin with a lowercase letter. “::” can be used as a namespace separator.

The same rules should be used when naming defined resource types, modules, and parameters, although modules and parameters cannot use the namespace separator.

Variable names can include alphanumeric characters and underscores, and are case-sensitive.

20. How do I document my manifests?

The puppet language includes a simple documentation syntax, which is currently documented on the Puppet Manifest Documentation wiki page. The puppetdoc command uses this inline documentation to automatically generate RDoc or HTML documents for your manifests and modules.

21. How do I manage passwords on Red Hat Enterprise Linux, CentOS, and Fedora Core?

As described in the Type reference, you need the Shadow Password Library, which is provided by the ruby-shadow package. The ruby-shadow library is available natively for fc6 (and higher), and should build on the corresponding RHEL and CentOS variants.

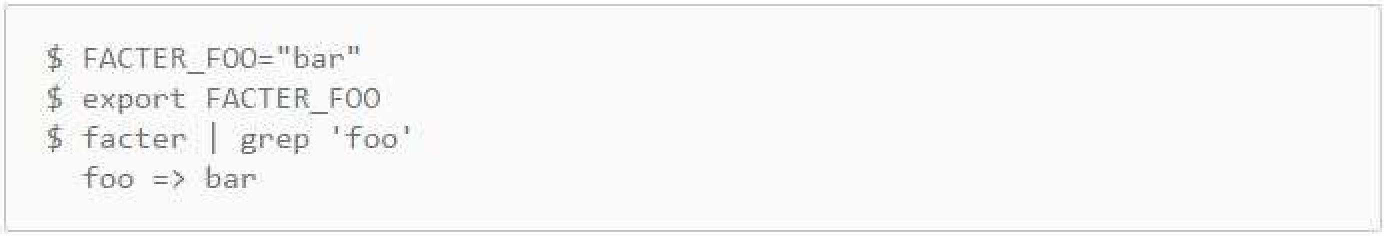
22. How do all of these variables, like $operatingsystem, get set?

The variables are all set by Facter. You can get a full listing of the available variables and their values by running facter by itself in a shell.

```
# facter
```

23. Can I access environment variables with Facter?

Not directly. However, Facter reads in custom facts from a special subset of environment variables. Any environment variable with a prefix of `FACTER_` will be converted into a fact when Facter runs. For example:



```
$ FACTER_FOO="bar"
$ export FACTER_FOO
$ facter | grep 'foo'
  foo => bar
```

The value of the `FACTER_FOO` environment variable would now be available in your Puppet manifests as $foo, and would have a value of 'bar'. Using shell scripting to export an arbitrary subset of environment variables as facts is left as an exercise for the reader.

24. Why shouldn't I use autosign for all my clients?

It is very tempting to enable autosign for all nodes, as it cuts down on the manual steps required to bootstrap a new node (or indeed to move it to a new puppet master).

Typically this would be done with a *.example.com or even * in the autosign.conf file.

This however can be very dangerous as it can enable a node to masquerade as another node, and get the configuration intended for that node. The reason for this is that the node chooses the certificate common name ('CN' - usually its fqdn, but this is fully configurable), and the puppet master then uses this CN to look up the node definition to serve. The certificate itself is stored, so two nodes could not connect with the same CN (eg alice.example.com), but this is not the problem.

The problem lies in the fact that the puppet master does not make a 1-1 mapping between a node and the first certificate it saw for it, and hence multiple certificates can map to the same node, for example:

- alice.example.com connects, gets node alice { } definition.
- bob.example.com connects with CN alice.bob.example.com, and also matches node alice { } definition.

Without autosigning, it would be apparent that bob was trying to get alice's configuration – as the puppet cert process lists the full fqdn/CN presented. With autosign turned on, bob silently retrieves alice's configuration.

Depending on your environment, this may not present a significant risk. It essentially boils down to the question 'Do I trust everything that can connect to my puppet master?'.

If you do still choose to have a permanent, or semi-permanent, permissive autosign.conf, please consider doing the following:

- Firewall your puppet master – restrict port tcp/8140 to only networks that you trust.
- Create puppet masters for each 'trust zone', and only include the trusted nodes in that Puppet masters manifest.
- Never use a full wildcard such as *.

25. What happens if I am on Puppet 2.6x or earlier?

Nothing changes for you. Puppet 2.6.x remains licensed as GPLv2. The license change is not retroactive.

26. Does this change affect all the components of Puppet?

As part of this change, we're also changing the license of the Facter system inventory tool to Apache. This change will take effect with Facter version 1.6.0, and earlier versions of Facter will remain licensed under the GPLv2 license. This change will bring the licensing of Puppet's two key components into alignment.

Our other major product, MCollective, is already licensed under the Apache 2.0 license.

27. What does this mean if I or my company have or want to contribute code to Puppet?

As part of this license change, Puppet Labs has approached every existing contributor to the project and asked them to sign a Contributor License Agreement or CLA.

Signing this CLA for yourself or your company provides both you and Puppet Labs with additional legal protections, and confirms:

1. That you own and are entitled to the code you are contributing to Puppet
2. That you are willing to have it used in distributions

This gives assurance that the origins and ownership of the code cannot be disputed in the event of any legal challenge.

28. What if I haven't signed a CLA?

If you haven't signed a CLA, then we can't yet accept your code contribution into Puppet or Facter. Signing a CLA is very easy: simply log into your GitHub account and go to our CLA page to sign the agreement.

We've worked hard to try find to everyone who has contributed code to Puppet, but if you have questions or concerns about a previous contribution you've made to Puppet and you don't believed you've signed a CLA, please sign a CLA or contact us for further information.

29. Does signing a CLA change who owns Puppet?

The change in license and the requirement for a CLA doesn't change who owns the code. This is a pure license agreement and NOT a Copyright assignment. If you sign a CLA, you maintain full copyright to your own code and are merely providing a license to Puppet Labs to use your code.

All other code remains the copyright of Puppet Labs.

30. Which versions of Ruby does Puppet support?

Puppet requires an MRI Ruby interpreter. Certain versions of Ruby are tested more thoroughly with Puppet than others, and some versions are not tested at all. Run ruby –version to check the version of Ruby on your system.

Starting with Puppet 4, puppet-agent packages do not rely on the OS’s Ruby version, as it bundles its own Ruby environment. You can install puppet-agent alongside any version of Ruby or on systems without Ruby installed. Likewise Puppet Enterprise does not rely on the OS’s Ruby version, as it bundles its own Ruby environment. You can install PE alongside any version of Ruby or on systems without Ruby installed. The Windows installers provided by Puppet Labs don’t rely on the OS’s Ruby version, and can be installed alongside any version of Ruby or on systems without Ruby installed.

31. What is Puppet ?

Puppet is a configuration Tool which is use to automate administration tasks.Puppet Agent(Client) sends request to Puppet Master (Server) and Puppet Master Push Configuration on Agent.

32. What is Manifests ?

Manifests, in Puppet, are the files in which the client configuration is specified.

33. What is Module and How it is different from Manifest ?

Whatever the manifests we defined in modules, can call or include into other manifests. Which makes easier management of Manifests.It helps you to push specific manifests on specific Node or Agent.

34. Command to check requests of Certificates ?

```
puppetca –list (2.6)
puppet ca list (3.0)
```

35. Command to sign Requested Certificates

```
puppetca  –sign hostname-of-agent (2.6)
puppet ca  sign hostname-of-agent (3.0)
```

36. Where Puppet Master Stores Certificates

```
/var/lib/puppet/ssl/ca/signed
```

37. What is Facter ?

Sometime you need to write manifests on conditional expression based on agent specific data which is available through Facter. Facter provides information like Kernel version, Dist release, IP Address, CPU info and etc. You can defined your facter also.

38. What is the use of etckeeper-commit-post and etckeeper-commit-pre on Puppet Agent ?

etckeeper-commit-post: In this configuration file you can define command and scripts which executes after pushing configuration on Agent
Etckeeper-commit-pre: In this configuration file you can define command and scripts which executes before pushing configuration on Agent

39. What is Puppet Kick ?

By default Puppet Agent request to Puppet Master after a periodic time which known as “runinterval”. Puppet Kick is a utility which allows you to trigger Puppet Agent from Puppet Master.

40. What is MCollective ?

MCollective is a powerful orchestration framework. Run actions on thousands of servers simultaneously, using existing plugins or writing your own.

CHEF INTERVIEW QUESTIONS

Can you explain briefly what Chef is?
Opscode Chef is an open-source systems integration framework built specifically for automating at scale. Using Ruby-based 'recipes' and 'cookbooks' of code commands, Chef makes it easy to deploy servers and scale applications throughout an entire infrastructure. Through a combination of configuration management and service-oriented architectures, Chef makes it easy to create fully automated infrastructure, while simplifying

systems management. Chef is available as an **open source download**, a **SaaS subscription**, or as **software installed behind the user's firewall**.

What's different about Chef compared to other automation solutions such as Puppet?
Puppet provides a declarative model for systems administration. It works well in small, mostly static environments, with low complexity. Opscode Chef provides a more flexible automation framework that allows enterprises to model their current workflow, at any scale, from development through to deployment and operations. Chef is based on primitives that create patterns that can be bent to any workflow/environment. Puppet is based on declaratives that can't be changed.

You recently released Chef 11. What are the major enhancements compared to the previous version?
Chef 11 was re-written from the ground up and leverages best-of-breed infrastructure technologies including the **Erlang programming language** and **PostgreSQL** database, delivering a rock-solid automation platform that can easily scale up to 10,000 nodes from a single Chef server – which is far greater than any previous Chef generation. Opscode is also two tiers of **commercial support** for open source Chef users (who are running Chef 11) covering both live system support and **cookbook** code troubleshooting. Other enhancements include:

- **Comprehensive Testing:** Chef 11 features the Pedant Testing Suite, delivering robust testing capabilities that can be executed with a single command, automating more than 2,000 end-to-end tests that ensure the Chef server is installed and working properly.
- **Easy Installation:** Chef 11 comes packaged with a 'one-click' installer, enabling easy and rapid deployment of Chef regardless of IT environment.
- **Enhanced Windows Support:** With the Pedant Testing Suite, Chef 11 includes automated testing across seven different versions of Windows, improving functionality and integration within Windows environments.

What resources (blogs, webinars, events) do you provide to get a solid technical understanding of Chef?
The most visible resource is the open source **Chef Community**, which is an important, active and vibrant online community where users can find recipes and cookbooks for everything from Windows to Hadoop, as well as a wide range of best practices, instruction guides, and more. However, the most important and

helpful resource is likely our **Documents page**, where users can find out everything they need to know about Chef, from getting started, to basic deployments, to advanced use cases, recipes, cookbooks, patches and more. We're working hard to make our Documents page your one-stop-shop for all things Opscode Chef.

Who owns Chef?
Chef is an open source systems integration framework stewarded and licensed by Opscode.

Which language is Chef written in?
The back-end API is written in Erlang. The front-end is Ruby.

How is Chef licensed?
As a free, open source download, a SaaS solution, or as enterprise software installed behind the firewall. The latter two are commercial solutions sold through a subscription model.

How many contributors / commits do you count?
Over 1,000 individual contributors, 170+ corporate contributors and tens of thousands of registered users. Open Source Chef has been downloaded nearly 2 million times in less than four years of availability.

Which functionalities is the community particularly focused on?
That's a tough question to answer because the Chef Community is so diverse, active and large that there are few "core focus" points. That said, ensuring Chef works seamlessly with MySQL, Apache and the many public cloud providers are frequent topics of conversation and code contribution, as is Chef + Windows.

Can you give us examples of typical use cases for Chef?
We call it the "three C's of Chef": Configuration Management of servers in physical data centers, private and public clouds; Continuous Application Delivery in any environment; and Cloud Management for public, private and hybrid clouds. The vast majority of use cases for any of the three flavors of Chef fall into one or more of these categories. We have solution pages on each of these use case available here, as well as a wide range of customer success stories available here.

Which prerequisites should enterprises meet when interested in using Chef?
There are no specific prerequisites needed to use Chef. However, familiarity with Ruby within IT and your Dev teams will be helpful, as is a willingness to deploy infrastructure as code for greater agility and less risk.

Tell us about your collaboration with Dell and Chef integration in Crowbar – Dell's deployment mechanism for OpenStack.
We're excited that Dell has embedded Opscode Chef in Crowbar and are very appreciative of the patches and cookbooks Dell has contributed back to the Chef Community. Dell has been a great partner in the OpenStack project and we look forward to more collaboration in the future.

Top AWS Architect Interview Questions

1. What is the difference between scalability and elasticity?
Scalability is the ability of a system to increase the workload on its current hardware resources to handle variability in demand. Elasticity is the ability of a system to increase the workload on its current and additional hardware resources, thereby enabling businesses to meet demand without investing in infrastructure up-front.

2. What are the different layers of cloud computing?
The three layers are:

- Infrastructure as a Service (IaaS)
- Platform as a Service (PaaS)
- Software as a Service (SaaS)

3. How to secure your data for transport in cloud?
Ensure that no one can intercept the data as it moves from point A to point B in the cloud and also check that there are no data leaks with the encryption key from any storage in the cloud. You can also segregate your data from other companies' data and then encrypt it by using an approved method. In addition you can ensure the security of older data that remains with a cloud vendor after you have no use for it.

4. List out different layers which define cloud architecture?
There are five layers:

- Cloud Controller (CLC)

- Walrus
- Cluster Controller
- Storage Controller (SC)
- Node Controller (NC)

5. What are the security laws which are implemented to secure data in a cloud?

The security laws which are implemented to secure data in cloud are:

- Processing
- File
- Output reconciliation
- Input Validation
- Security and Backup

6. What uses do API's have in cloud services?

Application Programming Interface (API) has the following uses:

- It eliminates the need to write fully fledged programs
- It provides the instructions to set up communication between one or more applications
- It allows easy creation of applications and links the cloud services with other systems

7. How many data centers are deployed for cloud computing? What are they?

There are two datacenters in cloud computing:

- Containerized Datacenters
- Low Density Datacenters

8. What is S3? What is it used for? Should encryption be used in S3?

According to Amazon, S3 is storage for the Internet. They define it as a, "simple storage service that offers software developers a highly-scalable, reliable, and low-latency data storage infrastructure at very low costs".

Amazon S3 provides a simple web service interface which you can use to store and retrieve any amount of data, at any time, from anywhere on the web. Using this web service, developers can easily build applications that make use of Internet storage.

Encryption should be considered for sensitive data, as S3 is a proprietary technology developed by Amazon themselves, and yet to be proven from a security standpoint.

9. What is Amazon EC2 service?
Amazon describes Elastic Compute Cloud (Amazon EC2) as a web service that provides resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers. Amazon EC2's simple web service interface allows developers to obtain and configure capacity with minimal friction.

10. What is an AMI?
An Amazon Machine Image (AMI) provides the information required to launch an instance, which is a virtual server in the cloud. You specify an AMI when you launch an instance, and you can launch as many instances from the AMI as you need. You can also launch instances from as many different AMIs as you need.
Source: http://docs.aws.amazon.com
An AMI includes the following:

- A template for the root volume for the instance (such as an operating system, an application server, and applications)
- Launch permissions that control which AWS accounts can use the AMI to launch instances
- A block device mapping that specifies the volumes to attach to the instance when it's launched

11. What is the relation between Instance and AMI?
An Amazon Machine Image (AMI) is a template that contains a software configuration (for example, an operating system, an application server, and applications). From an AMI, you launch an instance, which is a copy of the AMI running as a virtual server in the cloud.

You can launch different types of instances from a single AMI. An instance type determines the hardware of the host computer used for your instance. Each instance type offers different compute and memory capabilities.

12. What automation tools can you use to spinup servers?
Any of the following tools can be used:

- Roll-your-own scripts, and use the AWS API tools. Such scripts could be written in bash, perl or other language or your choice.
- Use a configuration management and provisioning tool like puppet or its successor Opscode Chef. You can also use a tool like Scalr.
- Use a managed solution such as Rightscale.

13. What are the different deployment models for Cloud?

The different models are:

- Private Cloud
- Public Cloud
- Hybrid Clouds

14. What is auto-scaling? How does it work?

Autoscaling is a feature of AWS which allows you to configure and automatically provision and spinup new instances without the need for your intervention. You can do this by setting thresholds and metrics to monitor. When those thresholds are crossed, a new instance of your choosing will be spun up, configured, and rolled into the load balancer pool.

15. What are the Security Best Practices for Amazon EC2?

There are several best practices for secure Amazon EC2. A few of them are given below:

- Use AWS Identity and Access Management (IAM) to control access to your AWS resources.
- Restrict access by only allowing trusted hosts or networks to access ports on your instance.
- Review the rules in your security groups regularly, and ensure that you apply the principle of least
- Privilege – only open up permissions that you require.
- Disable password-based logins for instances launched from your AMI. Passwords can be found or cracked, and are a security risk.

16. How is buffer used in Amazon web services?

Buffer is used to make the system more resilient to burst of traffic or load by synchronizing different components. The components always receive and process the requests in an unbalanced way. Buffer keeps the balance between different components and makes them work at the same speed to provide faster services.

17. What is the function of Amazon Elastic Compute Cloud?
Amazon Elastic compute cloud also known as Amazon EC2 is an Amazon web service that provides scalable resources and makes the computing easier for developers. The main functions of Amazon EC2 are:

- It provides easy configurable options and allow user to configure the capacity.
- It provides the complete control of computing resources and let the user run the computing environment according to his requirements.
- It provides a fast way to run the instances and quickly book the system hence reducing the overall time.
- It provides scalability to the resources and changes its environment according to the requirement of the user.
- It provides varieties of tools to the developers to build failure resilient applications.

18. What are the different components used in AWS?
The components that are used in AWS are:

- Amazon S3: it is used to retrieve input data sets that are involved in making a cloud architecture and also used to store the output data sets that is the result of the input.
- Amazon SQS: it is used for buffering requests that is received by the controller of the Amazon. It is the component that is used for communication between different controllers.
- Amazon SimpleDB: it is used to store intermediate status log and the tasks that are performed by the user/
- Amazon EC2: it is used to run a large distributed processing on the Hadoop cluster. It provides automatic parallelization and job scheduling.

19. Explain Stopping, Starting, and Terminating an Amazon EC2 instance?

- Stopping and Starting an instance: When an instance is stopped, the instance performs a normal shutdown and then transitions to a stopped state. All of its Amazon EBS volumes remain attached, and you can start the instance again at a later time. You are not charged for additional instance hours while the instance is in a stopped state.

- Terminating an instance: When an instance is terminated, the instance performs a normal shutdown, then the attached Amazon EBS volumes are deleted unless the volume’s deleteOnTermination attribute is set to false. The instance itself is also deleted, and you can’t start the instance again at a later time.

1. Explain Elastic Block Storage? What type of performance can you expect? How do you back it up? How do you improve performance?

EBS is a virtualized SAN or storage area network. That means it is RAID storage to start with so it's redundant and fault tolerant. If disks die in that RAID you don't lose data. Great! It is also virtualized, so you can provision and allocate storage, and attach it to your server with various API calls. No calling the storage expert and asking him or her to run specialized commands from the hardware vendor.

Performance on EBS can exhibit variability. That is it can go above the SLA performance level, then drop below it. The SLA provides you with an average disk I/O rate you can expect. This can frustrate some folks especially performance experts who expect reliable and consistent disk throughput on a server. Traditional physically hosted servers behave that way. Virtual AWS instances do not.

Backup EBS volumes by using the snapshot facility via API call or via a GUI interface like elasticfox.

Improve performance by using Linux software raid and striping across four volumes.

2. What is S3? What is it used for? Should encryption be used?

S3 stands for Simple Storage Service. You can think of it like ftp storage, where you can move files to and from there, but not mount it like a filesystem. AWS automatically puts your snapshots there, as well as AMIs there. Encryption should be considered for sensitive data, as S3 is a proprietary technology developed by Amazon themselves, and as yet unproven vis-a-vis a security standpoint.

3. What is an AMI? How do I build one?

AMI stands for Amazon Machine Image. It is effectively a snapshot of the root filesystem. Commodity hardware servers have a bios that points the the master boot record of the first block on a disk. A disk image though can sit anywhere physically on a disk, so Linux can boot from an arbitrary location on the EBS storage network.

Build a new AMI by first spinning up and instance from a trusted AMI. Then adding packages and components as required. Be wary of putting sensitive data onto an AMI. For instance your access credentials should be added to an instance after spinup. With a database, mount an outside volume that holds your MySQL data after spinup as well.

4. Can I vertically scale an Amazon instance? How?

Yes. This is an incredible feature of AWS and cloud virtualization. Spinup a new larger instance than the one you are currently running. Pause that instance and detach the root ebs volume from this server and discard. Then stop your live instance, detach its root volume. Note the unique device ID and attach that root volume to your new server. And the start it again. Voila you have scaled vertically in-place!!

5. What is auto-scaling? How does it work?

Autoscaling is a feature of AWS which allows you to configure and automatically provision and spinup new instances without the need for your intervention. You do this by setting thresholds and metrics to monitor. When those thresholds are crossed a new instance of your choosing will be spun up, configured, and rolled into the load balancer pool. Voila you've scaled horizontally without any operator intervention!

6. What automation tools can I use to spinup servers?

The most obvious way is to roll-your-own scripts, and use the AWS API tools. Such scripts could be written in bash, perl or other language or your choice. Next option is to use a configuration management and provisioning tool like puppet or better it's successor Opscode Chef. You might also look towards a tool like Scalr. Lastly you can go with a managed solution such as Rightscale.

7. What is configuration management? Why would I want to use it with cloud provisioning of resources?

Configuration management has been around for a long time in web operations and systems administration. Yet the cultural popularity of it has been limited. Most systems administrators configure machines as software was developed before version control - that is manually making changes on servers. Each server can then and usually is slightly
different. Troubleshooting though is straightforward as you login to the box and operate on it directly. Configuration

management brings a large automation tool into the picture, managing servers like strings of a puppet. This forces standardization, best practices, and reproducibility as all configs are versioned and managed. It also introduces a new way of working which is the biggest hurdle to its adoption.

Enter the cloud, and configuration management becomes even more critical. That's because virtual servers such as amazons EC2 instances are much less reliable than physical ones. You absolutely need a mechanism to rebuild them as-is at any moment. This pushes best practices like automation, reproducibility and disaster recovery into center stage.

8. Explain how you would simulate perimeter security using Amazon Web Services model?

Traditional perimeter security that we're already familiar with using firewalls and so forth is not supported in the Amazon EC2 world. AWS supports security groups. One can create a security group for a jump box with ssh access - only port 22 open. From there a webserver group and database group are created. The webserver group allows 80 and 443 from the world, but port 22 *only* from the jump box group. Further the database group allows port 3306 from the webserver group and port 22 from the jump box group. Add any machines to the webserver group and they can all hit the database. No one from the world can, and no one can directly ssh to any of your boxes.

Want to further lock this configuration down? Only allow ssh access from specific IP addresses on your network, or allow just your subnet.

1) Explain what is AWS?

AWS stands for Amazon Web Service; it is a collection of remote computing services also known as cloud computing platform. This new realm of cloud computing is also known as IaaS or Infrastructure as a Service.

2) Mention what are the key components of AWS?

The key components of AWS are

- **Route 53:** A DNS web service
- **Simple E-mail Service:** It allows sending e-mail using RESTFUL API call or via regular SMTP
- **Identity and Access Management:** It provides enhanced security and identity management for your AWS account
- **Simple Storage Device or (S3):** It is a storage device and the most widely used AWS service
- **Elastic Compute Cloud (EC2):** It provides on-demand computing resources for hosting applications. It is very useful in case of unpredictable workloads
- **Elastic Block Store (EBS):** It provides persistent storage volumes that attach to EC2 to allow you to persist data past the lifespan of a single EC2
- **CloudWatch:** To monitor AWS resources, It allows administrators to view and collect key Also, one can set a notification alarm in case of trouble.

3) Explain what is S3?

S3 stands for Simple Storage Service. You can use S3 interface to store and retrieve any amount of data, at any time and from anywhere on the web. For S3, the payment model is “pay as you go”.

4) Explain what is AMI?

AMI stands for Amazon Machine Image. It's a template that provides the information (an operating system, an application server and applications) required to launch an instance, which is a copy of the AMI running as a virtual server in the cloud. You can launch instances from as many different AMIs as you need.

5) Mention what is the relation between an instance and AMI?

From a single AMI, you can launch multiple types of instances. An instance type defines the hardware of the host computer used for your instance. Each instance type provides different compute and memory capabilities. Once you launch an instance, it looks like a traditional host, and we can interact with it as we would with any computer.

6) What does an AMI include?

An AMI includes the following things

- A template for the root volume for the instance
- Launch permissions decide which AWS accounts can avail the AMI to launch instances
- A block device mapping that determines the volumes to attach to the instance when it is launched

7) How can you send request to Amazon S3?

Amazon S3 is a REST service, you can send request by using the REST API or the AWS SDK wrapper libraries that wrap the underlying Amazon S3 REST API.

8) Mention what is the difference between Amazon S3 and EC2?

The difference between EC2 and Amazon S3 is that

EC2	S3
• It is a cloud web service used for hosting your application	• It is a data storage system where any amount of data can be stored
• It is like a huge computer machine which can run either Linux or Windows and can handle application like PHP, Python, Apache or any databases	• It has a REST interface and uses secure HMAC-SHA1 authentication keys

9) How many buckets can you create in AWS by default?

By default, you can create upto 100 buckets in each of your AWS accounts.

10) Explain can you vertically scale an Amazon instance? How?

Yes, you can vertically scale on Amazon instance. For that

- Spin up a new larger instance than the one you are currently running
- Pause that instance and detach the root webs volume from the server and discard
- Then stop your live instance and detach its root volume
- Note the unique device ID and attach that root volume to your new server
- And start it again

11) Explain what is T2 instances?

T2 instances are designed to provide moderate baseline performance and the capability to burst to higher performance as required by workload.

12) In VPC with private and public subnets, database servers should ideally be launched into which subnet?

With private and public subnets in VPC, database servers should ideally launch into private subnets.

13) Mention what are the security best practices for Amazon EC2?

For secure Amazon EC2 best practices, follow the following steps

- Use AWS identity and access management to control access to your AWS resources
- Restrict access by allowing only trusted hosts or networks to access ports on your instance
- Review the rules in your security groups regularly
- Only open up permissions that your require
- Disable password-based login, for instance, launched from your AMI

14) Explain how the buffer is used in Amazon web services?

The buffer is used to make the system more robust to manage traffic or load by synchronizing different component. Usually, components receive and process the requests in an unbalanced way, With the help of buffer, the components will be balanced and will work at the same speed to provide faster services.

15) While connecting to your instance what are the possible connection issues one might face?

The possible connection errors one might encounter while connecting instances are

- Connection timed out
- User key not recognized by the server
- Host key not found, permission denied
- Unprotected private key file
- Server refused our key or No supported authentication method available

- Error using MindTerm on Safari Browser
- Error using Mac OS X RDP Client

01. What is Amazon EC2 service?

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable (scalable) computing capacity in the cloud. You can use Amazon EC2 to launch as many virtual servers you need. In Amazon EC2 you can configure security and networking, and manage storage.

02. What are the features of Amazon EC2 service?

As Amazon EC2 service is a cloud service so it has all the cloud features. Amazon EC2 provides the following features:

- Virtual computing environment (known as instances)
- Pre-configured templates for your instances (known as Amazon Machine Images – AMIs)
- Amazon Machine Images (AMIs) is package that you need for your server (including the operating system and additional software)
- Amazon EC2 provides various configuration of CPU, memory, storage and networking capacity for your instances (known as instance type)
- Secure login information for your instances using key pairs (AWS stores the public key and you store the private key in a secure place)
- Storage volumes for temporary data that’s deleted when you stop or terminate your instance (known as instance store volumes)
- Amazon EC2 provides persistent storage volumes (using Amazon Elastic Block Store – EBS)
- A firewall that enables you to specify the protocols, ports, and source IP ranges that can reach your instances using security groups
- Static IP addresses for dynamic cloud computing (known as Elastic IP address)
- Amazon EC2 provides metadata (known as tags)

- Amazon EC2 provides virtual networks that are logically isolated from the rest of the AWS cloud, and that you can optionally connect to your own network (known as virtual private clouds – VPCs)

03. What is Amazon Machine Image and what is the relation between Instance and AMI?

Amazon Web Services provides several ways to access Amazon EC2, like web-based interface, AWS Command Line Interface (CLI) and Amazon Tools for Windows Powershell. First you need to signed up for an AWS account and you can access Amazon EC2.

Amazon EC2 provides a Query API. These requests are HTTP or HTTPS requests that use the HTTP verbs GET or POST and a Query parameter named Action.

04. What is Amazon Machine Image (AMI)?

An Amazon Machine Image (AMI) is a template that contains a software configuration (for example, an operating system, an application server, and applications). From an AMI, we launch an instance, which is a copy of the AMI running as a virtual server in the cloud. We can launch multiple instances of an AMI.

05. What is the relation between Instance and AMI?

We can launch different types of instances from a single AMI. An instance type essentially determines the hardware of the host computer used for your instance. Each instance type offers different compute and memory capabilities.

After we launch an instance, it looks like a traditional host, and we can interact with it as we would any computer. We have complete control of our instances; we can use sudo to run commands that require root privileges.

06. Explain storage for Amazon EC2 instance.

Amazon EC2 provides many data storage options for your instances. Each option has a unique combination of performance and durability. These storage can be used independently or in combination to suit your requirements.

There are mainly four types of storage provided by AWS.

Amazon EBS: Its durable, block-level storage volumes that you can attach to a running Amazon EC2 instance. The Amazon EBS volume persists independently from the running life of an Amazon EC2 instance. After an EBS volume is attached to an instance, you can use it like any other physical hard drive. Amazon EBS encryption feature supports encryption feature.
Amazon EC2 Instance Store: Storage disk that is attached to the host computer is referred to as instance store. Instance storage provides temporary block-level storage for Amazon EC2 instances. The data on an instance store volume persists only during the life of the associated Amazon EC2 instance; if you stop or terminate an instance, any data on instance store volumes is lost.
Amazon S3: Amazon S3 provides access to reliable and inexpensive data storage infrastructure. It is designed to make web-scale computing easier by enabling you to store and retrieve any amount of data, at any time, from within Amazon EC2 or anywhere on the web.
Adding Storage: Every time you launch an instance from an AMI, a root storage device is created for that instance. The root storage device contains all the information necessary to boot the instance. You can specify storage volumes in addition to the root device volume when you create an AMI or launch an instance using block device mapping.

07. What are the Security Best Practices for Amazon EC2?

There are several best practices for secure Amazon EC2. Following are few of them.

- Use AWS Identity and Access Management (IAM) to control access to your AWS resources.
- Restrict access by only allowing trusted hosts or networks to access ports on your instance.
- Review the rules in your security groups regularly, and ensure that you apply the principle of least
- Privilege — only open up permissions that you require.
- Disable password-based logins for instances launched from your AMI. Passwords can be found or cracked, and are a security risk.

08. Explain Stopping, Starting, and Terminating an Amazon EC2 instance?

Stopping and Starting an instance: When an instance is stopped, the instance performs a normal shutdown and then transitions to a stopped state. All of its Amazon EBS volumes remain attached, and you can start the instance again at a later time. You are not charged for additional instance hours while the instance is in a stopped state.
Terminating an instance: When an instance is terminated, the instance performs a normal shutdown, then the attached Amazon EBS volumes are deleted unless the volume's deleteOnTermination attribute is set to false. The instance itself is also deleted, and you can't start the instance again at a later time.

09. What are regions and availability zones in Amazon EC2? Explain in brief.

Amazon EC2 is hosted in multiple locations world-wide. These locations are composed of regions and Availability Zones. Each region is a separate geographic area. Each region has multiple, isolated locations known as Availability Zones.

Each region is completely independent. Each Availability Zone is isolated, but the Availability Zones in a region are connected through low-latency links. The following diagram illustrates the relationship between regions and Availability Zones.

10. Explain how to Launch EC2 instance in an Availability Zone?

Each region is completely independent and each Availability Zone is isolated. When you view your resources, you'll only see the resources tied to the region you have specified.

To launch a EC2 instance, you must select an AMI that's in the same region (if the AMI is in another region then you can copy the AMI to the region you are using). Now select an Availability Zone or let AWS choose for you. After creating the EC2 instance, it will show up in selected Availability Zone.

11. How to Migrate an Instance to another Availability Zone?

You can migrate your EC2 instance from one Availability Zone to another. Following are the steps to migrate an Instance to another Availability Zone.

1. Create an AMI from the running instance
2. Launch an instance from the AMI that you just created, specify the new Availability Zone
3. You can use the same instance type as the original instance, or select a new instance type
4. If the original instance has an associated Elastic IP address, then associate it with the new instance
5. If the original instance is a Reserved Instance, change the Availability Zone for your reservation

12. What is Amazon EC2 Root Device Volume?

When you launch an instance, the Root Device Volume contains the image used to boot the instance.
You can launch an instance from one of two types of AMIs:

1. Instance store-backed AMI
2. EBS based storage

13. How to persist Root Device Volume in Amazon EC2 Instance?

By default, the root device volume for an AMI backed by Amazon EBS is deleted when the instance terminates. To change the default behavior, set the DeleteOnTermination attribute to false using a block device mapping.

To change the root device volume of an instance to persist at launch using the console

1. Open the Amazon EC2 console.
2. From the Amazon EC2 console dashboard, click Launch Instance.
3. On the Choose an Amazon Machine Image (AMI) page, choose the AMI to use and click Select.
4. Follow the wizard to complete the Choose an Instance Type and Configure Instance Details pages.
5. On the Add Storage page, deselect the Delete On Termination check box for the root volume.
6. Complete the remaining wizard pages, and then click Launch.

Changing the Root Volume of an Instance to Persist Using the AWS CLI

Use the run-instances command to preserve the root volume by including a block device mapping that sets its DeleteOnTermination attribute for to false.

```
$ aws ec2 run-instances –image-id ami-1a2b3c4d –block-device-
mappings
```

```
`[{"DeviceName":"/dev/sda1","Ebs":{"DeleteOnTermination":false}}
]`
```

14. What is Key Pair?

AWS uses public-key cryptography to secure the login information for your instance. A Linux instance has no password; you use a key pair to log in to your instance securely.
You specify the name of the key pair when you launch your instance, then provide the private key when you log in using SSH.

15. How to create Key Pair?

We can create one using the Amazon EC2 console. To launch instances in multiple regions, we'll need to create a key pair in each region.
Following are the steps to create Key Pair:

1. Sign in to Amaon Web Service.
2. From the AWS dashboard, choose EC2 to open the Amazon EC2 console.
3. From the navigation bar, select a region for the key pair.
4. In the left navigation pane, under **NETWORK & SECURITY**, click Key **Pairs**.
5. Click Create **Key Pair**.
6. Enter a name for the new key pair in the **Key pair name** field of the **Create Key Pair** dialog box, and then click **Create**.
7. The private key file is automatically downloaded by your browser. The base file name is the name you specified as the name of your key pair, and the file name extension is .pem.

16. What is the use of Key Pair?

Key pair is used to log in to your instance securely. This is public-key cryptography to secure the login information for your instance.

17. What is Security Group in Amazon EC2?

Security groups act as a firewall for associated instances, controlling both inbound and outbound traffic at the instance level.

18. What are the features of Security Group in Amazon EC2?

Following are the features of the Security Group in Amazon EC2:

1. We can add rules to a security group that enable us to connect to our instance from our IP address using SSH.
2. We can also add rules that allow inbound and outbound HTTP and HTTPS access from anywhere.

19. How to create Security Group in Amazon EC2?

We can create Security Group in Amazon EC2 using the Amazon EC2 console. To launch instances in multiple regions, we'll need to create a Security Group in each region.

Following are the steps to create Security Group in Amazon EC2:

1. Open the Amazon EC2 console.
2. From the left navigation bar, select a region for the security group.
3. Click **Security Groups** in the navigation pane.
4. Click **Create Security Group**.
5. Enter a name for the new security group and a description.
6. In the **VPC** list, select your VPC.
7. On the **Inbound** tab, click **Add Rule** for each new rule, and then click **Create**.

20. How to launch an Amazon EC2 Instance?

We can launch Linux/Windows Amazon EC2 instance using AWS Management Console. Following are the steps to create Amazon EC2 instance.

1. Open the Amazon EC2 console.
2. From the console dashboard, choose Launch Instance.
3. Choose an Amazon Machine Image (AMI).
4. Choose an Instance Type.
5. Click on Review and Launch to let the wizard complete the other configuration setting.
6. On the Review Instance Launch page, under Security Groups select a Security Group.
7. Click on Launch on the Review Instance Launch.
8. Select an Existing ket pair when it prompte for key pair.
9. Click on View Instance to return on the console to see instance is launching.

21. How to connect to your Amazon EC2 Instance?

There are several ways to connect to a Linux instance. One of the commonly used method is to connect Linux instance from Windows local machine using PuTTY.

Following are the steps to connect to a Linux instance.

1. Install PuTTY on your local machine.
2. Get your instance ID.
3. Get the public DNS name of the instance.
4. Locate the private key.
5. Enable inbound SSH traffic from your IP address to your instance.
6. Converting Your Private Key Using PuTTYgen.
7. Starting a PuTTY Session.
8. Now you are connected to your EC2 instance.

22. How to add a EBS Volume to your Amazon EC2 Instance?

We can attach an EBS volume to one of our instances that is in the same Availability Zone as the Volume.

Following are the steps to attache an EBS volumn to an instance using console:

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the left navigation pane, choose **Volumes**.
3. Select a volume and choose **Attach Volume**.
4. Select the instance to which you want to attach the volume.
5. Click on **Attach**.
6. Now connect to your instance and make the volume available.

23. How to clean up your Amazon EC2 Instance and Volume?

After we are finished with the instance we created, we can clean up by terminating the instance.

Following are the steps to terminate the EC2 instance:

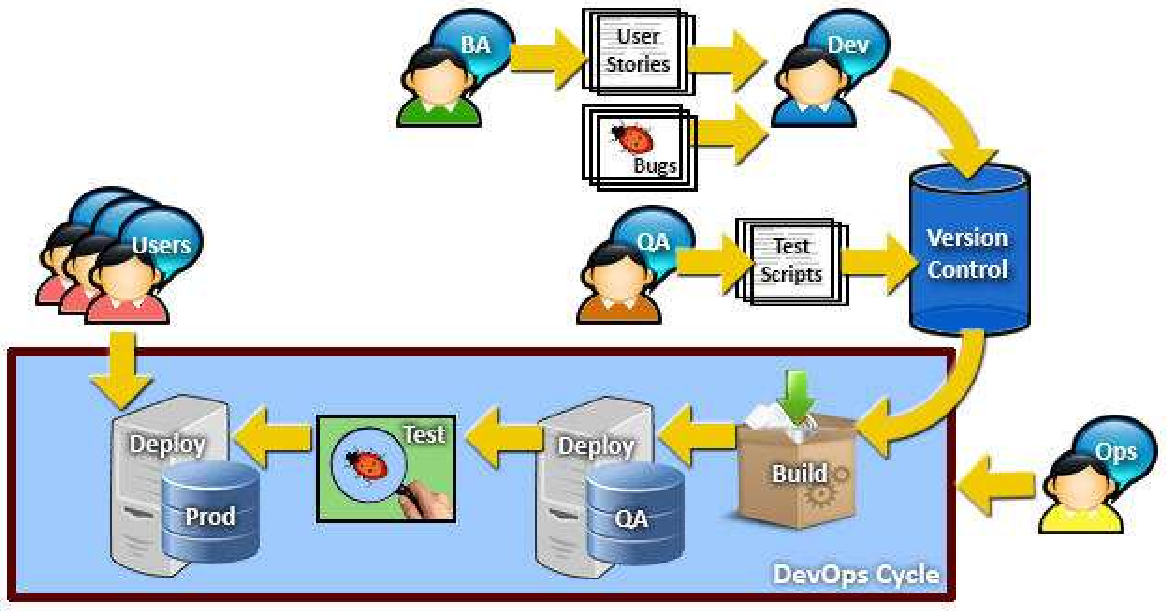
1. In the navigation pane, choose Instances. In the list of instances, select the instance.
2. Choose Actions, then Instance State, and then choose Terminate.
3. Choose Yes, Terminate when prompted for confirmation.

24. What are the best practices for Amazon EC2?

To get the maximum benefit from and satisfaction with Amazon EC2. There are mainly four best practices.

1. Security and Network Best Practices
2. Storage
3. Resource Management
4. Backup and Recovery

Cons	Pros	
• Slow-ish to respond and address customer concerns • Ruby-based, performance questionable compared to Python-based CM tools • Soon all customers must learn the Puppet DSL	• Mature solution • Good GUI • Support for all major OS's • Easy install	**Puppet**
• Still very new; not yet tried and tested by many • No support for Windows • GUI a work in progress	• Excellent performance, agent-less install and deploy • Low overhead, playbook based • Based on ubiquitous Python language • CLI accepts commands in almost any language	**Ansible**



DevOps Cycle © SoftwareTestingHelp.com

So an ideal DevOps cycle would start from:

- The dev writing code
- Building & deploying of binaries on a QA environment
- Executing test cases and finally
- Deploying on to Production in one smooth integrated flow.

HTTP

What is HTTP?

HTTP, the Hypertext Transfer Protocol, is the application-level protocol that is used to transfer data on the Web. HTTP comprises the rules by which Web browsers and servers exchange information. Although most people think of HTTP only in the context of the World-Wide Web, it can be, and is, used for other purposes, such as distributed object management systems.

How Does HTTP Work?

HTTP Is a request-response protocol. For example, a Web browser initiates a request to a server, typically by opening a TCP/IP connection. The request itself comprises

o a request line,
o a set of request headers, and
o an entity.

The server sends a response that comprises

o a status line,
o a set of response headers, and
o an entity.

The entity in the request or response can be thought of simply as the payload, which may be binary data. The other items are readable ASCII characters. When the response has been completed, either the browser or the server may terminate the TCP/IP connection, or the browser can send another request.

An Example

As an illustration of HTTP, here is an example exchange between a Web browser and the Silicon Press server, www.silicon-press.com. In response to a user request to go to the URL

http://www.silicon.press.com

the browser sends the following HTTP request to www.silicon-press.com:

```
GET / HTTP/1.1

Connection: Keep-Alive

User-Agent: Mozilla/5.0 (compatible; Konqueror/2.2-11; Linux)
```

```
Accept: text/*, image/jpeg, image/png, image/*, */*
Accept-Encoding: x-gzip, gzip, identity
Accept-Charset: Any, utf-8, *
Accept-Language: en, en_US
Host: www.silicon-press.com
-- blank line –
```

A brief explanation:

- o The first line is the request line that comprises three fields:
  1. a method: The GET method indicates that the server is supposed to return an entity.
  2. a request-URI (Universal Resource Identifier). The / indicates the root of the document system on the server, and
  3. HTTP protocol version: 1.1 in this case.
- o The second line is the optional Connection header informs the server that the browser would like to leave the connection open after the response.
- o The third line is the optional User-Agent header that identifies the kind of browser that is sending the request, its version, and its operating system.
- o The Accept headers specify the type, language, and encoding for the returned entity that the browser would prefer to receive from the server.

Responding to the browser, the www.silicon-press.com server sends the following response:

```
HTTP/1.1 200 OK
Date: Thu, 24 Jan 2002 17:33:52 GMT
Server: Apache/1.3.14
Last-Modified: Mon, 21 Jan 2002 22:08:33 GMT
Etag: “47bc6-25e0-3c4c9161”
Accept-Ranges: bytes
Content-Length: 9696
Connection: close
Content-Type: text/html
-- blank line--
-- HTML entity --
```

A brief explanation:

- o The first line is the status line consisting of three fields:
  1. HTTP protocol version of the response: 1.1 in this case,
  2. a three-digit numeric status code, and

   3. a short description of the status code.

- The Content-Length, Content-Type, Etag, and Last-Modified header lines describe the entity returned.

HTTPS (Secure HTTP)

HTTPS denotes the use of HTTP with SSL (Secure Socket Layer) protocol or its successor protocol Transport Layer Security (TLS), a transport-layer protocol. Either of these protocols, which use encryption, can be used to create a secure connection between two machines. The browser uses SSL or TLS when connecting to a secure part of a website indicated by an HTTPS URL, that is, a URL with the prefix https://. The browser then uses HTTP to send and receive requests over this secure connection.

Top 60 Linux Technical Term Full Form

S.No	**Technical Term**	**Full Form**
1	UUID	Universally Unique Identifier library.
2	NFS	Network File System.
3 .	SMB	Server Message Block.
4	SWAT	Samba Web Administration Tool.
5	CIFS	Common Internet File System.
6	PXE boot	Preboot Execution Enviroment
7	GD	Graphical Draw.

8	YUM	Yellowdog Updater Modified.
9	RPM	RPM Package Manager.
10	GCC	GNU Compiler Collection.
11	NRPE	Nagios Remote Plugin Executor.
12	WWN	World Wide Name.
13	WWID	World Wide Identifier.
14	SGID	Set Group ID.
15	SUID	Set User ID.
16	URL	Uniform Resource Locator.
17	FSCK	File System Check.
18	GNU	Gnu s Not Unix (operating system)
19	GNOME	GNU Network Object Model Environment
20	KDE	K Desktop Environment

21	Proc	Process information pseudo-filesystem.
22	NIS	Network Information Systems.
23	LDAP	Lightweight Directory Access Protocol.
24	FTP	File Transfer Protocol.
25	SFTP Protocol.	Secure File Transfer
26	SSH	Secure Shell.
27	IMAP	Internet message access protocol
28	POP3	Post Office Protocol
29	SMTP	Simple Mail Transfer Protocol
30	SSL	Secure Socket Layer
31	IP	Internet Protocol.
32	**HTTP**	Hyper Text Transfer Protocol.
33	MKFS	Make Filesystem.
34	LS	list Directory.

35	PWD	Present Working Directory.
36	**SED**	stream Editor.
37	CHOWN	Change Ownership
38	CAT	Concatenate files.
39	MKDIR	Make Directories.
40	RMDIR	Remove Empty Directory.
41	**NETSTAT**	Network Statistics.
42	DU	Disk Usage.
43	HWCLOCK	Hardware Clock.
44	env	Environment Variable.
45	DNS	Domain Name System.
46	SNMP	Simple Network Management Protocol.
47	ps	Current Process.
48	Inode	Index Node.
49	Lsattr	List file attributes.

50	Chattr	Change file attributes.
51	IDS	Intrusion detection system.
52	Grep	Globally search a regular expression and print.
53	egrep .	Extended Global Regular Expression Print [Unix]
54	Bash	Bourne Again Shell
55	Tcsh	TENEX C shell
56	Lvm	Logical Volume Manager
57	NTP	Network Time Protocol
58	Tcp	Transmission Control Protocol
59	Udp	User Datagram Protocol.
60	DRBD	Distributed Replicated Block Device.

Vim Cheat Sheet

Cursor movement

- `h` - move cursor left
- `j` - move cursor down
- `k` - move cursor up
- `l` - move cursor right
- `w` - jump forwards to the start of a word
- `W` - jump forwards to the start of a word (words can contain punctuation)
- `e` - jump forwards to the end of a word
- `E` - jump forwards to the end of a word (words can contain punctuation)
- `b` - jump backwards to the start of a word
- `B` - jump backwards to the start of a word (words can contain punctuation)
- `0` - jump to the start of the line
- `^` - jump to the first non-blank character of the line
- `$` - jump to the end of the line
- `g_` - jump to the last non-blank character of the line
- `gg` - go to the first line of the document
- `G` - go to the last line of the document
- `5G` - go to line 5
- `fx` - jump to next occurrence of character x
- `tx` - jump to before next occurrence of character x
- `}` - jump to next paragraph (or function/block, when editing code)
- `{` - jump to previous paragraph (or function/block, when editing code)
- `Ctrl` + `b` - move back one full screen
- `Ctrl` + `f` - move forward one full screen
- `Ctrl` + `d` - move forward 1/2 a screen
- `Ctrl` + `u` - move back 1/2 a screen

Tip Prefix a cursor movement command with a number to repeat it. For example, `4j`moves down 4 lines.

Insert mode - inserting/appending text

- `i` - insert before the cursor
- `I` - insert at the beginning of the line
- `a` - insert (append) after the cursor
- `A` - insert (append) at the end of the line
- `o` - append (open) a new line below the current line
- `O` - append (open) a new line above the current line
- `ea` - insert (append) at the end of the word
- `Esc` - exit insert mode

Editing

- `r` - replace a single character
- `J` - join line below to the current one
- `cc` - change (replace) entire line
- `cw` - change (replace) to the end of the word
- `c$` - change (replace) to the end of the line
- `s` - delete character and substitute text
- `S` - delete line and substitute text (same as cc)
- `xp` - transpose two letters (delete and paste)
- `u` - undo
- `Ctrl` + `r` - redo
- `.` - repeat last command

Marking text (visual mode)

- `v` - start visual mode, mark lines, then do a command (like y-yank)
- `V` - start linewise visual mode
- `o` - move to other end of marked area
- `Ctrl` + `v` - start visual block mode
- `O` - move to other corner of block
- `aw` - mark a word
- `ab` - a block with ()
- `aB` - a block with {}
- `ib` - inner block with ()
- `iB` - inner block with {}
- `Esc` - exit visual mode

Visual commands

- `>` - shift text right
- `<` - shift text left
- `y` - yank (copy) marked text
- `d` - delete marked text
- `~` - switch case

Cut and paste

- `yy` - yank (copy) a line
- `2yy` - yank (copy) 2 lines
- `yw` - yank (copy) word
- `y$` - yank (copy) to end of line
- `p` - put (paste) the clipboard after cursor
- `P` - put (paste) before cursor
- `dd` - delete (cut) a line

- `2dd` - delete (cut) 2 lines
- `dw` - delete (cut) word
- `D` - delete (cut) to the end of the line
- `d$` - delete (cut) to the end of the line
- `x` - delete (cut) character

Exiting

- `:w` - write (save) the file, but don't exit
- `:w !sudo tee %` - write out the current file using sudo
- `:wq` or `:x` or `ZZ` - write (save) and quit
- `:q` - quit (fails if there are unsaved changes)
- `:q!` or `ZQ` - quit and throw away unsaved changes

Search and replace

- `/pattern` - search for pattern
- `?pattern` - search backward for pattern
- `\vpattern` - 'very magic' pattern: non-alphanumeric characters are interpreted as special regex symbols (no escaping needed)
- `n` - repeat search in same direction
- `N` - repeat search in opposite direction
- `:%s/old/new/g` - replace all old with new throughout file
- `:%s/old/new/gc` - replace all old with new throughout file with confirmations
- `:noh` - remove highlighting of search matches

Search in multiple files

- `:vimgrep /pattern/ {file}` - search for pattern in multiple files

e.g. `:vimgrep /foo/ **/*`

- `:cn` - jump to the next match
- `:cp` - jump to the previous match
- `:copen` - open a window containing the list of matches

Working with multiple files

- `:e` filename - edit a file in a new buffer
- `:bnext` or `:bn` - go to the next buffer
- `:bprev` or `:bp` - go to the previous buffer
- `:bd` - delete a buffer (close a file)
- `:ls` - list all open buffers
- `:sp` filename - open a file in a new buffer and split window

- `:vsp` filename - open a file in a new buffer and vertically split window
- `Ctrl` + `ws` - split window
- `Ctrl` + `ww` - switch windows
- `Ctrl` + `wq` - quit a window
- `Ctrl` + `wv` - split window vertically
- `Ctrl` + `wh` - move cursor to the left window (vertical split)
- `Ctrl` + `wl` - move cursor to the right window (vertical split)
- `Ctrl` + `wj` - move cursor to the window below (horizontal split)
- `Ctrl` + `wk` - move cursor to the window above (horizontal split)

Tabs

- `:tabnew` filename or `:tabn` filename - open a file in a new tab
- `Ctrl` + `wT` - move the current split window into its own tab
- `gt` or `:tabnext` or `:tabn` - move to the next tab
- `gT` or `:tabprev` or `:tabp` - move to the previous tab
- `#gt` - move to tab number #
- `:tabmove #` - move current tab to the #th position (indexed from 0)
- `:tabclose` or `:tabc` - close the current tab and all its windows
- `:tabonly` or `:tabo` - close all tabs except for the current one
- `:tabdo` command - run the `command` on all tabs (e.g. `:tabdo q` - closes all opened tabs)

Systemvorraussetzungen

Erfolgreiche Installation von Puppet: puppet, puppet-commo
Unix oder Linux artiges Betriebssystem oder Windows hier l
und kein puppetmaster Server installieren.

Systemwerte (Fakten) ermitteln

```
facter
```

Gibt eine Liste aller bekannten Systemwerte zurück auf welche in PP Skripten zugegriffen werden kann.

Puppet Skripte ausführen

```
puppet apply install-java.pp
```

Führt das im aktuellen Verzeichnis liegende Puppetskript *install-java.pp* aus.

PP Skript – Definition der Ausführungspfade

```
Exec { path => [ "/bin/", "/sbin/" , "/usr/bin/",
```

Einige Puppet Module führen Shell Kommandos aus. Damit diese gefunden werden, müssen die üblichen Systempfade in Puppet bekannt sein.

Puppet Module verwalten

puppet module list	Alle installierten Mo
puppet module search java	Sucht installierbare M
puppet module uninstall puppetlabs-s-tdlib	Deinstalliert das Mo
puppet module install puppetlabs-stdlib	Installiert das Modul
puppet apply --modulepath ~/puppet--modules/modules/ script.pp	Apply script.pp wob modules/modules/ ve

Unter Ubuntu sollten alle Befehle und Kommandos über sudo mit root Rechten ausgeführt werden. z.B.:
sudo puppet module list

PP Skript – Konfiguration einer Klasse

```
class { 'eclipse':
  method => 'download',
  release_name => 'kepler',
  service_release => 'SR2',
  ensure => present,
}
```

Aufruf der Klasse eclipse (welche von einem Modul bereitgestellt wird) und Zuweisung benötigter Parameter.

PP Skript – Ausführen von Shell Kommandos

```
# stdlib
exec { "stdlib-modul":
    command => "sudo puppet module install --forc
    path => "/usr/local/bin/:/bin/:/usr/bin",
}
```

Das im Skript aufgerufene Shell Kommando installiert das stdlib Modul von puppetlabs.

PP Skript – Loggen von Ausgaben

```
# save current vars
file { "/tmp/facts.yaml":
    content => inline_template("<%= scope.to_hash
k.is_a?(String) && v.is_a?(String) ) }.to_yaml %>
}
```

Speichert die Ausgaben vom *facter* Kommando in die Datei /tmp/facts.yaml.
Hilfreich um herauszufinden mit welchen Fakten ein Puppet Skript ausgeführt wurde.

Module bauen und installieren

1. Es muss im Modulverzeichnis ein Modulefile existieren (re
2. Aus dem Parentfolder ruft man auf:
>puppet module build [moduledir]
3.
>puppet module install [moduledir]/pkg/[vendor]-[module]-[v
4. Test the module lokal
5. Upload to puppet forge via web upload as logged in user.

PP Skript – Einrichtung eines Nodes

```
node default {
include eclipse
notice('Well done!')
}
```

Die Klasse *eclipse* wird aufgerufen und alle damit verbundenen Aktionen durchgeführt – abhängig von der Konfiguration der Klasse kann das sowohl Installation als auch Deinstallationen sowie reine Konfigurationen beinhalten.

PP Skript – Einrichtung eines Services

```
tomcat::instance { 'tomcat8':
  catalina_base => '/opt/apache-tomcat/tomcat8',
  source_url => 'http://mirror.nexcess.net/apache
8/v8.0.15/bin/apache-tomcat-8.0.15.tar.gz'
}->
tomcat::service { 'default':
  catalina_base => '/opt/apache-tomcat/tomcat8',
}
```

Hier wird Tomcat als laufender Service einger-
ichtet.

Chef Cheat Sheet

General

Chef Dry Run

```
chef-client -Fmin --why-run
```

List Facts

```
ohai
```

Bootstrap Chef client

```
knife bootstrap <FQDN/IP>
```

Change Chef Run List

```
knife node run_list <add|remove> <node> <cookbook>::<recipe>
```

Runlist Status

```
knife status --run-list
knife status "role:webserver" --run-list
```

Nodes and Roles

List Node Info

```
knife node show <node>
```

List Nodes per Role

```
knife search node 'roles:<role name>'
```

Load role from file

```
knife role from file <file> [<file> [...]]
```

Data Bags

Load data bag from file

```
knife data bag from file <data bag name> <file>
```

knife + SSH

```
knife ssh -a ipaddress name:server1 "chef-client"
```

you can also use patterns:

```
knife ssh -a ipaddress name:www* "uptime"
```

Debugging

Inheritance

Debugging Attribute Inheritance

```
# Invoke chef shell in attribute mode
chef-shell -z
chef > attributes
chef:attributes >

# Query attributes examples
chef:attributes > default["authorized_keys"]
[...]
chef:attributes > node["packages"]
[...]
```

Editing Files

using a Script resource.

```
bash "some_commands" do
    user "root"
    cwd "/tmp"
    code <<-EOT
       echo "alias rm='rm -i'" >> /root/.bashrc
    EOT
end
```

- `knife job start ...`
- `knife job list`
-

```
knife node status ...
```

Getting Knife Version

```
knife --version
```

Create Cookbook

```
knife cookbook create <cookbookName>
```

Getting List of all the client nodes

```
knife client list
```

Server Bootstrap

```
knife bootstrap <hostname/ipaddr> -x <username> -P <password> -N <nodeName>
```

Server Bootstrap with Runlist

```
knife bootsrap <hostname> -x root -P <password> -N module3 -r "receipe[apache]"
```

Server Bootstrap with Sudo with providing user pem file as ssh identity with , and and additional passing to the node

```
knife bootstrap <hostname> --sudo -x <user> -i <SSH Itentityfile> --node-name <nodename> --run-list <runlist> -E <Environment> --no-host-key-verify -j <json_attribute>
```

Add Receipe to RunList for Node

```
knife node run_list add module2 "receipe[apache]"
```

Ohai Command

```
ohai
```

Knife Node Show

```
knife node show <nodename>
know node show <nodename> -a <keyName>
knife node show module2 -a apache
knife node show module2
```

Knife Remove Item from run_list

```
knife node run_list remove module2 "receipe[apache]"
```

***** Kitchen Commands *****

```
kitchen list
kitchen create
kitchen login <InstanceName>
```

Knife Configure

Knife Configure command is used to create knife.rb and client.rb so that they can distribute to workstation and nodes.

Configure client.rb

```
knife configure client <directory>
```

Configure knife.rb

```
knife configure
```

VAGRANT

Creating the Vagrantfile

vagrant init	Initialize Vagrant with a Vagrantfile and ./.vagra specified base image. Before you can do vagran base image in the Vagrantfile.
vagrant init boxpath	Initialize Vagrant with a specific box. To find a b find one you like, just replace it's name with bo: init chef/centos-6.5.

Vagrantfile customizations

```
vagrant.configure("2") do |config|
    config.vm.box = "chef/centos-6.5"
    # guest is the VM; host is your computer end
    config.vm.network "forwarded_port", guest: 80
    config.vm.provision :shell, path: "my_bash_sc
    # path is relative to your Vagrantfile
end
```

By default ./ on your computer is shared as /vagrant on the VM. Letting other people access your VM's

Boxes

vagrant box list	List the inst
vagrant box add <name> <box path/HTTP URI>	Add the bo
vagrant box remove <name> virtualbox	delete a bo

Common Vagrant Commands

vagrant up	starts vagrant environment (also provisions up) Equivalent to pressing the power butto
vagrant status	outputs status of the vagrant machine
vagrant halt	stops the vagrant machine
vagrant reload	restarts vagrant machine, loads new Vagra
vagrant provision	forces reprovisioning of the vagrant machi
vagrant ssh	connects to machine via SSH
vagrant destroy	stops and deletes all traces of the vagrant r
vagrant suspend	Suspends a virtual machine (remembers st
vagrant resume	Resume a suspended machine (vagrant up well)
vagrant reload --provision	Restart the virtual machine and force provi
vagrant provision --debug	Use the debug flag to increase the verbosit

Be sure that you are in the same directory as the Vagrantfile when running these commands!

vagrant box outdated	Check for u

Boxes are prebuilt VM images. You never modify your box images

Tips	
vagrant -v	Get the vagrant version
vagrant global-status	outputs status of all vagra
vagrant global-status --prune	same as above, but prune
vagrant push	Vagrant can be configured
vagrant up --provision | tee provision.log	Runs `vagrant up`, forces output to a file
VAGRANT_LOG=info vagrant up	Use the environement var verbosity

Containers

Your basic isolated Docker process. Containers are to Virtual Machines as threads are to processes. Or you can think of them as chroots on steroids.

Lifecycle

- docker create creates a container but does not start it.
- docker rename allows the container to be renamed.
- docker run creates and starts a container in one operation.
- docker rm deletes a container.
- docker update updates a container's resource limits.

If you want a transient container, docker run --rm will remove the container after it stops.
If you want to map a directory on the host to a docker container, docker run -v $HOSTDIR:$DOCKERDIR. Also see Volumes.
If you want to remove also the volumes associated with the container, the deletion of the container must include the -v switch like in docker rm -v.
There's also a logging driver available for individual containers in docker 1.10. To run docker with a custom log driver (i.e. to syslog), use docker run --log-driver=syslog

Starting and Stopping

- docker start starts a container so it is running.
- docker stop stops a running container.
- docker restart stops and starts a container.
- docker pause pauses a running container, "freezing" it in place.
- docker unpause will unpause a running container.
- docker wait blocks until running container stops.
- docker kill sends a SIGKILL to a running container.
- docker attach will connect to a running container.

If you want to integrate a container with a host process manager, start the daemon with -r=false then use docker start -a.
If you want to expose container ports through the host, see the exposing ports section.

Restart policies on crashed docker instances are covered here.

Info

- docker ps shows running containers.
- docker logs gets logs from container. (You can use a custom log driver, but logs is only available for json-file andjournald in 1.10)
- docker inspect looks at all the info on a container (including IP address).
- docker events gets events from container.
- docker port shows public facing port of container.
- docker top shows running processes in container.
- docker stats shows containers' resource usage statistics.
- docker diff shows changed files in the container's FS.

docker ps -a shows running and stopped containers.
docker stats --all shows a running list of containers.

Import / Export

- docker cp copies files or folders between a container and the local filesystem..
- docker export turns container filesystem into tarball archive stream to STDOUT.

Executing Commands

- docker exec to execute a command in container.

To enter a running container, attach a new shell process to a running container called foo, use: docker exec -it foo /bin/bash.

Images

Images are just templates for docker containers.

Lifecycle

- docker images shows all images.
- docker import creates an image from a tarball.
- docker build creates image from Dockerfile.
- docker commit creates image from a container, pausing it temporarily if it is running.
- docker rmi removes an image.
- docker load loads an image from a tar archive as STDIN, including images and tags (as of 0.7).
- docker save saves an image to a tar archive stream to STDOUT with all parent layers, tags & versions (as of 0.7).

Info

- docker history shows history of image.
- docker tag tags an image to a name (local or registry).

Cleaning up

While you can use the docker rmi command to remove specific images, there's a tool called docker-gc that will clean up images that are no longer used by any containers in a safe manner.

Networks

Docker has a networks feature. Not much is known about it, so this is a good place to expand the cheat sheet. There is a note saying that it's a good way to configure docker containers to talk to each other without using ports. See working with networks for more details.

Lifecycle

- docker network create
- docker network rm

Info

- docker network ls
- docker network inspect

Connection

- docker network connect
- docker network disconnect

You can specify a specific IP address for a container:

```
# create a new bridge network with your subnet and gateway for
your ip block
docker network create --subnet 203.0.113.0/24 --gateway
203.0.113.254 iptastic

# run a nginx container with a specific ip in that block
$ docker run --rm -it --net iptastic --ip 203.0.113.2 nginx
```

```
# curl the ip from any other place (assuming this is a public ip
block duh)
$ curl 203.0.113.2
```

Registry & Repository

A repository is a *hosted* collection of tagged images that together create the file system for a container.

A registry is a *host* -- a server that stores repositories and provides an HTTP API for managing the uploading and downloading of repositories.

Docker.com hosts its own index to a central registry which contains a large number of repositories. Having said that, the central docker registry does not do a good job of verifying images and should be avoided if you're worried about security.

- `docker login` to login to a registry.
- `docker logout` to logout from a registry.
- `docker search` searches registry for image.
- `docker pull` pulls an image from registry to local machine.
- `docker push` pushes an image to the registry from local machine.

Run local registry

You can run a local registry by using the docker distribution project and looking at the local deploy instructions.

SALT STACK

Minion Overview

```
salt-run manage.up                             # Shows what Minions are up

salt-run manage.down                           # Shows what Minions are down or
not connected

salt-run manage.status                         # Shows both online and offline
Minions

salt '*'' test.ping                            # Pings all minions
```

Targetting Minions

```
salt '*' some_module                           # target all Salt Minions

salt 'web*' some_module                        # Target Minion(s) based on their
Minion ID

salt -G 'oscodename:wheezy' some_module        # Target Minions based on their
grains
```

Job Management

```
salt-run jobs.list_jobs                        # Lists ALL Jobs

salt-call saltutil.running                     # Lists running jobs
```

```
salt-call saltutil.kill_job <job id number>  # Kills a specific running job

salt-run jobs.active                         # get list of active jobs

salt-run jobs.lookup_jid <job id number>     # get details of this specific job
```

States

```
salt-run state.highstate                     # runs all states targetted for a
minion, on a minion

salt 'ns*' state.highstate                   # runs all states targetted for a
minion, from master

salt 'web*' state.sls settings.nginx         # runs settings/nginx/init.sls on
web*
```

Grains

```
salt '*' grains.ls                           # List all grains on all minions

salt '*' grains.item os                      # Show the value of the OS grain
for every minion

salt '*' grains.item roles                   # Show the value of the roles
grain for every minion

salt 'minion1' grains.setval mygrain True    # Set mygrain to True (create if
it doesn't exist yet)
```

```
salt 'minion1' grains.delval mygrain         # Delete the value of the grain
```

Documentation on the system

```
salt '*' sys.doc                              # output sys.doc (= all
documentation)

salt '*' sys.doc pkg                          # only sys.doc for pkg module

salt '*' sys.doc network                      # only sys.doc for network module

salt '*' sys.doc system                       # only sys.doc for system module

salt '*' sys.doc status                       # only sys.doc for status module
```

Ansible Cheat Sheet

Playbooks

```
ansible-playbook <YAML>                 # Run on all hosts defined
ansible-playbook <YAML> -f 10           # Run 10 hosts parallel
ansible-playbook <YAML> --verbose             # Verbose on
successful tasks
ansible-playbook <YAML> -C                    # Test run
ansible-playbook <YAML> -C -D           # Dry run
ansible-playbook <YAML> -l <host>             # Run on single host

# Run Infos
ansible-playbook <YAML> --list-hosts
ansible-playbook <YAML> --list-tasks

# Syntax Check
```

```
ansible-playbook --syntax-check <YAML>
```

Remote Execution

```
ansible all -m ping

# Execute arbitrary commands
ansible <hostgroup> -a <command>
ansible all -a "ifconfig -a"
```

Debugging

List facts and state of a host

```
ansible <host> -m setup
ansible <host> -m setup -a 'filter=ansible_eth*'
```