MODULE 2

DevOps

LESSON

PRE-TEST

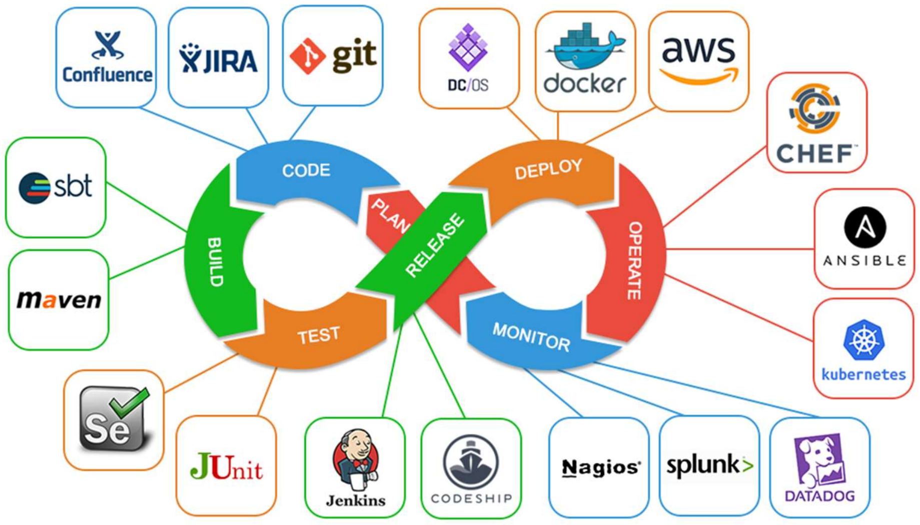
LESSON 4

DevOps Overview

LEARNING OUTCOMES

After studying this lesson, you should be able to:

1. Understand the history and meaning of DevOps.
2. Learn the benefits of DevOps



What Does the Term DevOps Mean?

The term DevOps is a blend of development and operations.

DevOps is a set of practices that works to automate and integrate the processes between software development and IT teams, so they can build, test, and release software faster and more reliably. The term DevOps was formed by combining the words "development" and "operations" and signifies a cultural shift that bridges the gap between development and operation teams, which historically functioned in siloes.

A compound of development (Dev) and operations (Ops), DevOps is the union of people, process, and technology to continually provide value to customers.

What does DevOps mean for teams?

DevOps enables formerly siloed roles—development, IT operations, quality engineering, and security—to coordinate and collaborate to produce better, more reliable products. By adopting a DevOps culture along with DevOps practices and tools, teams gain the ability to better respond to customer needs, increase confidence in the applications they build, and achieve business goals faster.

At its essence, DevOps is a culture, a movement, a philosophy.

It's a firm handshake between development and operations that emphasizes a shift in mindset, better collaboration, and tighter integration. It unites agile, git, continuous delivery, automation, and much more, to help development and operations teams be more efficient, innovate faster, and deliver higher value to businesses and customers.

"DevOps isn't any single person's job. It's everyone's job." - Christophe Capel
Principal Product Manager, Jira Service Desk

What DevOps Is NOT

The term DevOps is a slightly overloaded one. To understand the scope of the DevOps concept, it helps to discuss what DevOps is not. DevOps is not a marketing (buzz) term.

Although some aspects of DevOps are not new, it is a new and strong movement intended to improve the delivery process. The DevOps approach accepts the daily challenges in software delivery and provides steps to address them. DevOps does not allow developers to work on the production system. It is not a free "process" that opens production-relevant aspects to developers.

For example, DevOps does not grant developers permission to work on production systems. Instead, DevOps is about discipline, conventions, and a defined process that is transparent for all.

History of DevOps

How development and operations teams came together to solve dysfunction in the industry.

Despite the rise of agile methodology, development and operations teams remained siloed for years. DevOps is the next evolution of collaboration tools and practices to release better software, faster.

Bringing development and IT teams together

The DevOps movement started to coalesce some time between 2007 and

2008, when IT operations and software development communities raised concerns what they felt was a fatal level of dysfunction in the industry.

They railed against the traditional software development model, which called for those who write code to be organizationally and functionally apart from those who deploy and support that code.

Developers and IT/Ops professionals had separate (and often competing) objectives, separate department leadership, separate key performance indicators by which they were judged, and often worked on separate floors or even separate buildings. The result was siloed teams concerned only with their own fiefdoms, long hours, botched releases, and unhappy customers. Surely there's a better way, they said. So, the two communities came together and started talking – with people like Patrick Dubois, Gene Kim, and John Willis driving the conversation.

What began in online forums and local meet-ups is now a major theme in the software zeitgeist, which is probably what brought you here! You and your team are feeling the pain caused by siloed teams and broken lines of communication within your company.

You're using agile methodologies for planning and development, but still struggling to get that code out the door without a bunch of drama. You've probably heard a few things about DevOps and the seemingly magical effect it can have on teams: Nearly all (99%) of DevOps teams are confident about the success of their code that goes into production, in a survey of 500 DevOps practitioners conducted by Atlassian.

However, DevOps isn't magic, and transformations don't happen overnight. The good news is that you don't have to wait for upper management to roll out a large-scale initiative. By understanding the value of DevOps and making small, incremental changes, your team can embark on the DevOps journey right away.

Influences and Origins

Patrick Debois coined the term DevOps in 2009 while organizing the DevOpsDays conference in Belgium. This was the first in a series of relevant conferences dedicated to the concept that helped spread the popularity of the term. Many past movements, early adopters, and influences helped coin DevOps and transform DevOps into an accepted term:

- Patrick Debois ran a session called "Agile Operations and Infrastructure: How Infra-gile Are You?" at the Agile 2008 conference in Toronto and published a paper with a similar name.
- Marcel Wegermann published an e-mail list called "Agile System Administration."

- John Allspaw gave a presentation called "10+ Deploys per Day: Dev and Ops Cooperation"7 at the Velocity 2009 conference in San Jose.
- Steven Blank published a book called Four Steps to the Epiphany.
- Eric Ries published The Lean Startup and others have written on the "lean startup" scene.
- The 451 Group published the first analyst report on DevOps (titled "The Rise of DevOps") in September 2010.

Labeling tools or approaches as being aligned with "DevOps" without reflecting on the concrete content or without trying to define DevOps tends to result in random buzzwords. Thus, one may ask what is the motivation for the DevOps movement? To understand this motivation better, we'll now examine the underlying conflict between development and operations.

Going beyond agile

DevOps touches every phase of the development and operations lifecycle. From planning and building to monitoring and iterating, DevOps brings together the skills, processes, and tools from every facet of an engineering and IT organization.

Agile methodologies help teams plan and produce by breaking work down into manageable tasks and milestones. Agile relies on sprints, backlogs, epics, and stories to assign work to skilled team members adjust timelines when necessary, and deliver quality products and services to customers.

Continuous integration and delivery: Continuous integration and delivery is a cornerstone of DevOps practices that relies on automating the merging and deployment of code. Traditional development methods require engineers to manually update changes in the codebase, with additional manual checks to ensure quality code is ready to ship into production. Deployments are scheduled with weeks- or months-long delays to remove the likelihood of bugs or incidents. DevOps practices remove these delays by automating the merging, testing, and deployment functions. High-performing teams use CI/CD to reduce their deployment frequency from every few months to multiple times each day.

Git repositories and workflows enable the automation and version control capabilities that are foundational to DevOps practices. Because Git is distributed, operations such as commit, blame, diff, merge, and log happens faster. Git also supports branching, merging, and rewriting repository history, which enables powerful workflows and tools.

IT service management is the process IT teams use to manage the end-to-end delivery of IT services to customers. This includes all the processes and activities to design, create, deliver, and support IT services. The core concept of ITSM is the belief that IT should be delivered as a service, which goes beyond basic IT support. ITSM teams oversee all kinds of workplace technology, ranging from laptops to servers, to business-critical software applications.

Incident management teams respond to an unplanned event or service interruption and restore the service to its operational state. In a "you build it, you run it" model, developers partner with operations to reduce the likelihood of an incident occurring, and also reduce the meantime to recovery when an incident happens.

Benefits of DevOps

When development and operations teams come together, they reduce lead time, deploy more frequently, and produce higher-quality software.

The value of DevOps is big. Nearly all (99%) of respondents said DevOps has had a positive impact on their organization, according to the 2020 DevOps Trends Survey. Teams that practice DevOps ship better work faster, streamline incident responses and improve collaboration and communication across teams.

1. Collaboration and trust

Building a culture of shared responsibility, transparency, and faster feedback is the foundation of every high-performing DevOps team. Collaboration and problem-solving ranked as the most important elements of a successful DevOps culture, according to our 2020 DevOps Trends survey.

Teams that work in silos often don't adhere to the systems thinking DevOps espouses. Systems thinking is being aware of how your actions not only affect your team but all the other teams involved in the release process. Lack of visibility and shared goals means a lack of dependency planning, misaligned priorities, finger-pointing, and "not our problem" mentality, resulting in slower velocity and substandard quality. DevOps is that change in the mindset of looking at the development process holistically and breaking down the barrier between development and operations.

2. Release faster and work smarter

Speed is everything. Teams that practice DevOps release deliverables more frequently, with higher quality and stability. The DORA "2019 State of DevOps" report found that elite teams deploy 208 times more frequently and 106 times faster than low-performing teams.

A lack of automated test and review cycles slow the release to production, while poor incident response time kills velocity and team confidence. Disparate tools and processes increase operating costs, lead to context switching, and can slow down momentum. Yet with tools that drive automation and new processes, teams can increase productivity and release more frequently with fewer hiccups.

3. Accelerate time-to-resolution

The team with the fastest feedback loop is the team that thrives. Full transparency and seamless communication enable DevOps teams to minimize downtime and resolve issues faster.

If critical issues aren't resolved quickly, customer satisfaction tanks. Key issues slip through the cracks in the absence of open communication, resulting in increased tension and frustration among teams. Open communication helps development and operations teams swarm on issues, fix incidents, and unblock the release pipeline faster.

4. Better manage unplanned work

Unplanned work is a reality that every team faces–a reality that most often impacts team productivity. With established processes and clear prioritization, development and operations teams can better manage unplanned work while continuing to focus on planned work.

Transitioning and prioritizing unplanned work across different teams and systems is inefficient and distracts from work at hand. However, through raised visibility and proactive retrospection, teams can better anticipate and share unplanned work.

Teams who fully embrace DevOps practices work smarter and faster, and deliver better quality to their customers. The increased use of automation and cross-functional collaboration reduces complexity and errors, which in turn improves the Mean Time to Recovery (MTTR) when incidents and outages occur.

Development and Operations in Conflict

Traditional organizations divide their teams by type of work (that often results in what are called silos). Certain development departments specialize in writing code. Many companies also have dedicated departments for testing software. Because bringing software to production and maintaining it there often require skills other than software development, an operations department is created. Splitting work areas appears to benefit the management as well. In addition to the specialized team, each department has its own manager who fulfills the individual requirements needed for this specific department.

Each department defines its goals based on the division of labor. The development department may be measured by its speed in creating new features, whereas the operations department may be judged by server uptime and application response time. Unfortunately, operations is considered to be successful if the metrics are stable and unchanging, whereas development is only applauded if many things change. Because conflict is baked into this system, intensive collaboration is unlikely.
Development teams strive for change, whereas operations teams strive for stability. The conflict between development and operations is caused by a combination of conflicting motivations, processes, and tooling. As a result, isolated silos evolve (see Figure 1-1).

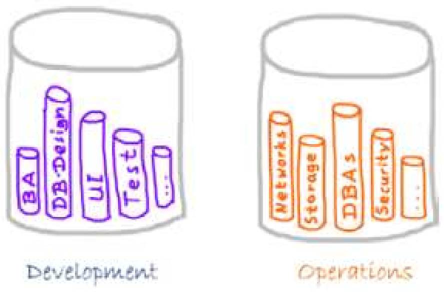


Figure 1-1. Development and operations are two distinct departments. Often, these departments act like silos because they are independent of each other 11

In a nutshell, the conflict between development and operations is as follows:

- Need for change: Development produces changes (e.g., new features, bug fixes, and work based on change requests). They want their changes rolled out to production.
- Fear of change: Once the software is delivered, the operations department wants to avoid making changes to the software to ensure stable conditions for the production systems.

LESSON 5

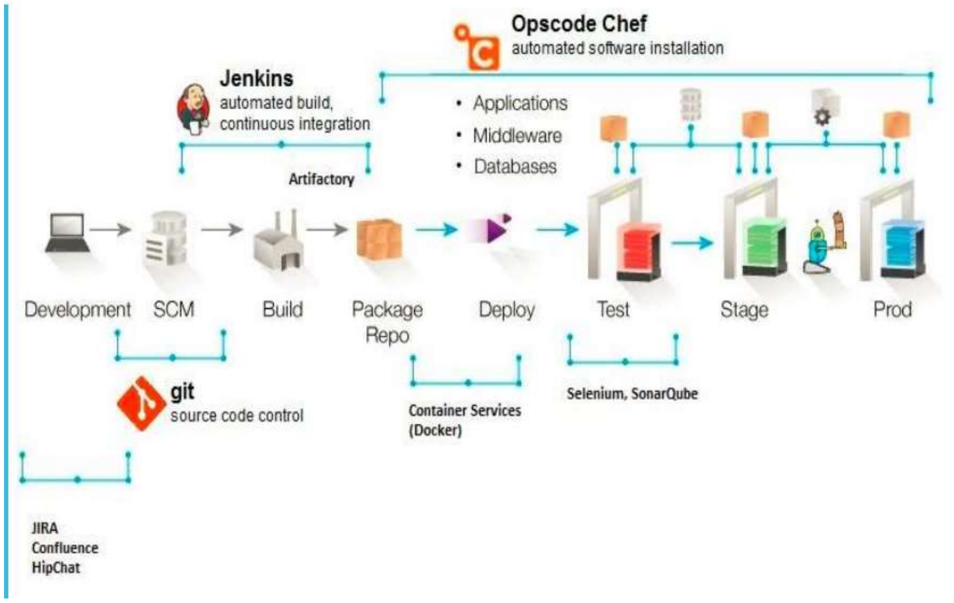
DevOps Architecture

LEARNING OUTCOMES

After studying this lesson, you should be able to:

1. Introduce the concept of DevOps.
2. Learn points to be considered while designing DevOps Architecture

DevOps Architecture



While designing DevOps Architecture the below points to be considered

Designers must thoroughly understand customer use cases:

- Usability, reliability, scaling, availability, testability, and supportability are more important than individual features! Quality over quantity is recommended. Designs that anticipate actual customer usage are the most successful.

The culture needs to support designers:

- Leaders must support designers with motivation, mentoring, and training. No designer can be expected to know everything. It’s OK for designers to make some mistakes if lessons are learned and improvement quickly follows. Good DevOps practices including continuous monitoring and quick remediation are examples of practices that help minimize the impact of any mistakes.

Supporting tools are needed to realize Design for DevOps practices:

- Elastic infrastructures that can be easily orchestrated, created and released as needed to support designers' tasks on-demand with minimal delay.

- Design, code management, monitoring, and test tools readily available and scalable with minimal delay.

- Monitoring tools that track application process performance and report the results to designers in easily consumable formats without delay.

Design practices should support QA:

- Understand the QA process.

- Software code changes are pre-checked with unit tests before commit to the integration/trunk branch.

- Software source code changes are pre-checked with Static Analysis tools before commit to the integration branch.

- Software code changes are pre-checked with dynamic analysis and regression tests before commit to the integration/trunk branch to ensure the software performance has not degraded.

Design practices should support Operations:

- Understand the delivery and deployment of pipeline processes.

- Software features are tagged with software switches (i.e. feature tags or toggles) during check-in to enable selective feature level testing, promotion, and reverts.

- Automated test cases are checked-in to the integration branch at the same time code changes are checked-in. Evidence that the tests passed is included with the check-in.

- Tests are conducted in a pre-flight test environment that is a close facsimile of the production environment.

- Products are architected to support modular independent packaging, testing, and releases.
- In other words, the product itself is partitioned into modules with minimal dependencies between modules. In this way the modules can be built, tested, and released without requiring the entire product to be built, tested, and released all at once.

- Where possible, applications are architected as modular, immutable microservices ready for deployment in cloud infrastructures, in accordance with the tenets of 12-factor immutable non-monolithic apps, rather than monolithic mutable architectures.

- Software code changes are pre-checked using peer code reviews before commit to the integration/trunk branch.

- Software changes are integrated into a private environment together with the most recent integration branch version and tested using functional testing before committing the software changes to the integration/trunk branch.

- Developers commit their code changes regularly, at least once per day.

LESSON 6

DevOps Tools

LEARNING OUTCOMES

After studying this lesson, you should be able to:

1. Know the different types of DevOps tools.

There are 9 types of DevOps tools which have known before choosing for the project.

Collaboration Tools:

- This type of tool is crucial to helping teams work together more easily, regardless of time zones or locations.
- Rapid action-oriented communication is designed to share knowledge and save time. (See: Slack, Campfire).

Planning Tools:

- This type of tool is designed to provide transparency to stakeholders and participants.
- Working together, teams can plan towards common goals and a better understanding of dependencies. Bottlenecks and conflicting priorities are more visible. (See: Clarizenand Asana).

Source Control Tools:

- Tools of this sort make up the building blocks for the entire process ranging across all key assets. Whether code, configuration, documentation, database, compiled resources, and your web site HTML – you can only gain by managing them in your one true source of truth. (See: Git, Subversion).

Issue Tracking Tools:

- These tools increase responsiveness and visibility.
- All teams should use the same issue tracking tool, unifying internal issue tracking as well as customer-generated ones. (See: Jira and ZenDesk).

Configuration Management Tools:

- Without this type of tool, it would be impossible to enforce desired state norms or achieve any sort of consistency at scale.
- Infrastructure should be treated exactly as a code that can be provisioned and configured in a repeatable way. (See: Puppet, Chef, Salt).

Database DevOps Tools:

- The database needs to be an honored member of the managed resources family. Managing source code, tasks,
- configuration, and deployments is incomplete if the database is left
- out of the equation. (See: DBmaestro)

Continuous Integration Tools:

- Continuous integration tools provide an immediate feedback loop by regularly merging code. Teams merge developed code many times a day, getting feedback from automated test tools. (See: Jenkins, Bamboo, TeamCity).

Automated Testing Tools:

- Tools of this sort are tasked with verifying code quality before passing the build. The quicker the feedback loop works – the higher the quality gets, and the quicker you reach the desired "definition of done". (See: Telerik, QTP, TestComplete)

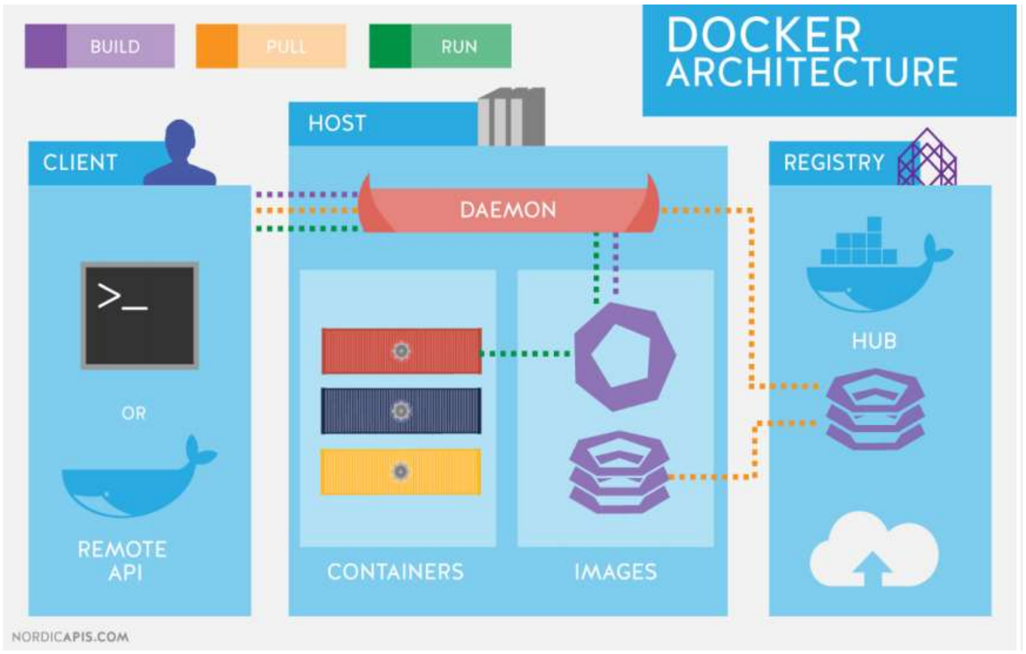
Deployment Tools:

- In an effective DevOps environment, application deployments are frequent, predictable, and reliable.
- Deployment tools are essential to checking those boxes. Continuous delivery means that applications can be released to production at any time you want in order to improve time to market, while keeping risk as low as possible. (See: IBM uDeploy, CA Release Automation, XebiaLabs)

Docker

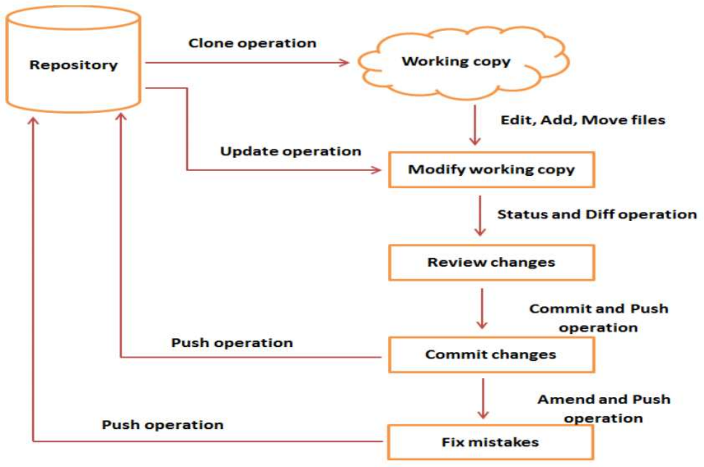
Docker is at the forefront of the new trend toward containerization. It packages together everything that an application needs to run—the code, the runtime, system tools, libraries, etc.—so that applications will operate the same way no matter where they are deployed. Containers are more lightweight than virtual machines, and they also offer some security benefits.

A recent survey conducted by Docker found that 80 percent of enterprises surveyed plan their DevOps implementations around Docker. Docker implements a high-levelAPI to provide lightweight containers that run processes in isolation.



Git

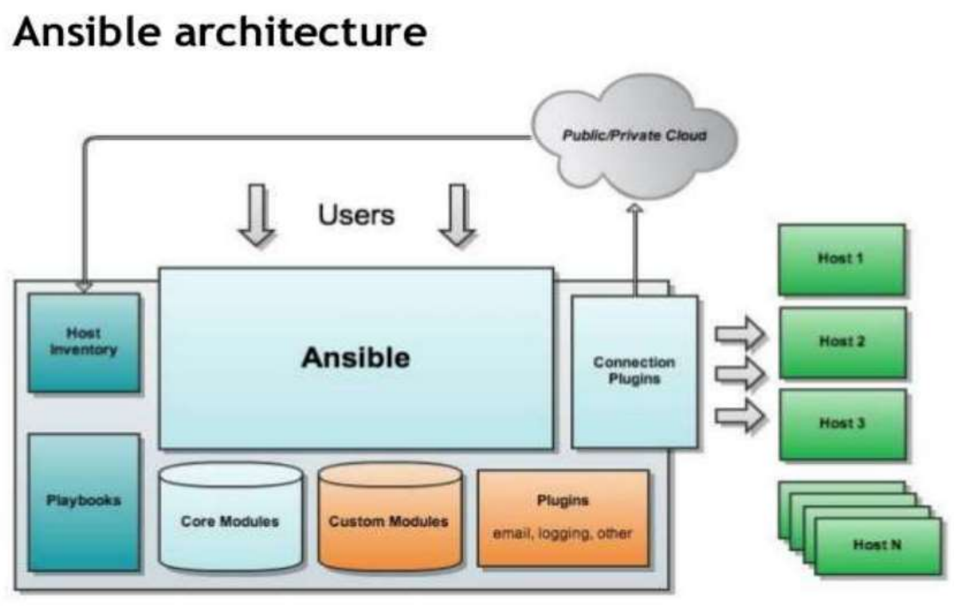
In recent years, Git has become incredibly popular for source code management, particularly as the site GitHub has become more popular for hosting open source projects. It stands out from other version control management for the ease with which it handles branching and merging. It's also very easy to use with distributed development teams, and it offers fast performance. Many DevOps teams use it to manage the source code for their applications. Its list of well-known users includes many of the biggest firms in the technology industry, such as Google, Facebook, Microsoft, Twitter, LinkedIn, Netflix, the Linux kernel and many others.



Ansible

Owned by Red Hat, Ansible automates many common IT operations tasks, such as cloud provisioning, configuration management and application deployment. It integrates with a lot of other popular DevOps tools, including Git, JIRA, Jenkins and many others. The software has been downloaded more than 5 million times, and it has more than sixteen thousand stars on GitHub. The free open source version is available on GitHub, and Red Hat offers three paid versions—self-support, standard and premium—with prices that vary based on the number of nodes in production and the level of support needed.

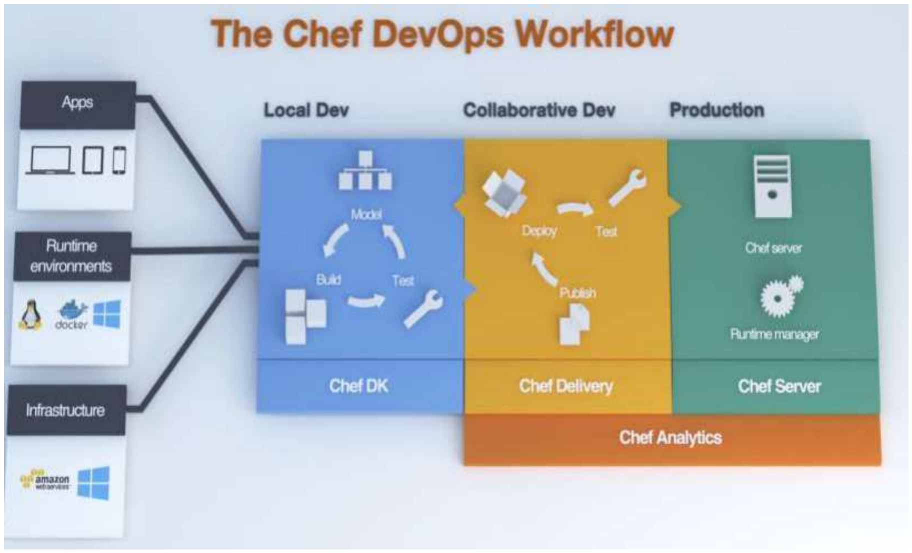
Ansible has two types of servers: controlling machines and nodes.
First, there is a single controlling machine which is where orchestration begins.



Chef

Another option for infrastructure automation, Chef makes it possible to manage both cloud and traditional environments with a single tool. It promises to accelerate cloud adoption while maintaining high availability. Quite a lot of documentation and technical resources are available on the Chef site, including many resources designed to help enterprises transition to DevOps and scale their DevOps implementations.

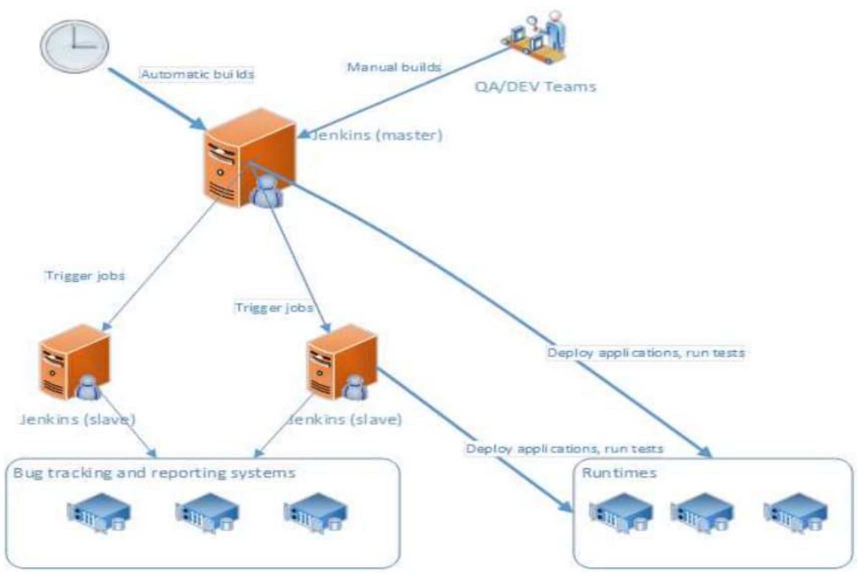
The company also offers a paid version of Chef called Chef Automate, as well as two other open source projects: InSpec, which focuses on security and compliance, and Habitat, which makes it possible to deploy apps in any environment, including the cloud, bare metal or containers.



Jenkins

The "leading open source automation server," Jenkins was forked from Hudson and offers many of the same capabilities. It boasts easy installation and configuration, hundreds of plugins, extensibility and a distributed architecture that allows it to speed the process of testing.

It has a very active user community with lots of scheduled events that offer opportunities to learn more about the software. There is also plenty of documentation on the website, including a blog that is updated regularly. Jenkins Pipeline is a suite of plugins which supports implementing and integrating continuous delivery pipelines into Jenkins. Pipeline provides an extensible set of tools for modeling simpleto-complex delivery pipelines "as code".



POST-TEST

MODULE 2:

TEST I:

1. **DevOps means.**
   A) Developers taking over all Operations tasks.
   B) Automating the process of software delivery and infrastructure changes.
   C) The collaboration and communication of both software developers and other information-technology (IT) professional while automating the process of software delivery and infrastructure changes.
   D) The collaboration and communication of just software developers and operations staff while automating the process software delivery and infrastructure changes.

2. **Is this statement correct? "DevOps is more than just a tool or a process change, it inherently requires an organizational culture shift".**
   A) Yes, there needs to be cultural shift within the organization across all stakeholders to ensure a successful adoption of a DevOps approach.
   B) Yes, but the most up to date tools and LEAN processes need to be in place to drive an organizational culture shift.
   C) No, DevOps is all about the tools.
   D) No, cultural shift will occur when staff are using the most up to date tools and LEAN processes.

3. **The adoption of DevOps is being driven by factors such as:**
   A) Use of Agile and other development processes and methodologies
   B) Demand for an increased rate of production releases from application and business unit stakeholders
   C) Wide availability of virtualized and cloud infrastructure from internal and external providers
   D) Increased usage of data Centre automation and configuration management tools

4. **Which benefits of adopting a DevOps approach could be included in a business case to adopt a DevOps approach?**
   A) Improved deployment frequency, which can lead to faster time to market
   B) Lower failure rate of new releases
   C) Shortened lead time between fixes
   D) Faster mean time to recovery in the event of a new release crashing or

otherwise disabling the current system

5. **Which statement best describes the relationship between DevOps and Continuous Delivery?**
   A) DevOps and Continuous Delivery are the same thing.
   B) DevOps and Continuous Delivery are not related and are mutually exclusive.
   C) DevOps and Continuous Delivery share a background in Agile methods and LEAN thinking.
   D) DevOps and Continuous Delivery share common processes.

6. **Agile and DevOps are similar but differ in a few important aspect. Which statement is correct?**
   A) Agile is a change of thinking whereas DevOps is actual organization cultural change
   B) Agile is actual organizational cultural change whereas DevOps is a change of thinking.
   C) Agile is process driven whereas DevOps is role driven.
   D) Agile is role driven whereas DevOps is process driven.

7. **Which of these statements are correct about DevOps?**
   A) DevOps and ITIL® don't mix
   B) DevOps won't work in regulated industries
   C) DevOps won't work with Outsourced Development
   D) You must use cloud technologies

8. **Which statement best describes the role of Change Management within a DevOps environment?**
   A) Nothing changes as a risk adverse Change Management approach is paramount to IT and business success.
   B) The moment an app change is asked for, the request should go to the Developers to authorise. Once authorised it goes to Operations for implementation.
   C) The moment an app change is asked for, the request should go out to everyone on the team, no matter which IT discipline they work in.
   D) DevOps does not need Change Management?

9. **Which statement best describes the goal of DevOps?**
   A) One goal of DevOps is to establish an environment where Change Management does bot control application releases.
   B) One goal of DevOps is to establish an environment where releasing more reliable applications faster and more frequently can occur.
   C) One goal of DevOps is to establish an environment where application development perform all operations tasks.
   D) One goal of DevOps is to establish an environment where releasing

applications is valued over the quality of the released application

10. The development teams that support the Agile approach to DevOps must include staff from the operations teams to ensure:
A) That stability is prioritised over creativity
B) Operational considerations are prioritised over stability
C) Operational considerations are taken into account
D) The resultant designs of the systems will fit nicely into the business as usual environment