<https://onecognizant.cognizant.com/?GlobalAppid=2173&url=https%3A%2F%2Fonecognizantapps.cognizant.com%2F2173%2FAssociate%2FCourseDetails%3FCourseCode%3Dos_devo_a01_it_enus>

[**https://library.skillport.com/courseware/Content/ria/RIA\_V3\_1\_121/index\_tablet.html?lang=en&AICC\_URL=https%3A%2F%2Fpvsp73zbe.skillport.com%3A443%2Fskillportbe%2Fspcognizant%2FResult.rbe&AICC\_SID=anil.mandava@cognizant.com-306855812-os\_devo\_a01\_it\_enus-A0-@0-SPCSF&CBTLAUNCH=os\_devo\_a01\_it\_enus\_c&COURSEINFO=skins/sp\_pt0pc1ssl1db0&DYNAMIC\_SKIN\_URL=https://pvsp73zbe.skillport.com:443/skillportbe/spcognizant/Cmd.be&tryRIA=true&use508=0&SIGNED\_APPLET=true**](https://library.skillport.com/courseware/Content/ria/RIA_V3_1_121/index_tablet.html?lang=en&AICC_URL=https%3A%2F%2Fpvsp73zbe.skillport.com%3A443%2Fskillportbe%2Fspcognizant%2FResult.rbe&AICC_SID=anil.mandava@cognizant.com-306855812-os_devo_a01_it_enus-A0-@0-SPCSF&CBTLAUNCH=os_devo_a01_it_enus_c&COURSEINFO=skins/sp_pt0pc1ssl1db0&DYNAMIC_SKIN_URL=https://pvsp73zbe.skillport.com:443/skillportbe/spcognizant/Cmd.be&tryRIA=true&use508=0&SIGNED_APPLET=true)

**DevOps Fundamentals: Tools, Technologies, and Infrastructures**

Top of Form



Bottom of Form

Top of Form



Bottom of Form

|  |
| --- |
| [Play This Course](javascript:var%20warnMsg%20=%20'You%20are%20about%20to%20access%20content%20that%20may%20not%20support%20web%20accessibility.%20Click%20OK%20to%20continue,%20or%20Cancel%20to%20abort.';flagPlayerCourse%20=%20true;launchApplet(lmsSessionTokenId,courseName,%20courseType,winParams,%20use508);disablePlayButton(1,%200);) |

[Overview/Description](https://cognizant.skillport.com/skillportfe/bcs/SinglePassUserCmd.cfm?sessionid=anil.mandava@cognizant.com-629434572&courseaction=launch&x508=0&x508defined=0&restype=0&callfromolsa=1&coursename=os_devo_a01_it_enus#C1)  
[Target Audience](https://cognizant.skillport.com/skillportfe/bcs/SinglePassUserCmd.cfm?sessionid=anil.mandava@cognizant.com-629434572&courseaction=launch&x508=0&x508defined=0&restype=0&callfromolsa=1&coursename=os_devo_a01_it_enus#C2)  
[Expected Duration](https://cognizant.skillport.com/skillportfe/bcs/SinglePassUserCmd.cfm?sessionid=anil.mandava@cognizant.com-629434572&courseaction=launch&x508=0&x508defined=0&restype=0&callfromolsa=1&coursename=os_devo_a01_it_enus#C3)  
[Prerequisites](https://cognizant.skillport.com/skillportfe/bcs/SinglePassUserCmd.cfm?sessionid=anil.mandava@cognizant.com-629434572&courseaction=launch&x508=0&x508defined=0&restype=0&callfromolsa=1&coursename=os_devo_a01_it_enus#C5)  
[Objectives](https://cognizant.skillport.com/skillportfe/bcs/SinglePassUserCmd.cfm?sessionid=anil.mandava@cognizant.com-629434572&courseaction=launch&x508=0&x508defined=0&restype=0&callfromolsa=1&coursename=os_devo_a01_it_enus#C4)

**Overview/Description**

DevOps is a software methodology where the operations and development staff participating together in the entire service lifecycle, from the design phase through to the development and production phases. In this course you will learn the fundamentals of the DevOps methodology, and the principals and processes of the DevOps workflow. You will also be introduced to various DevOps tools such as JIRA, Confluence, Bitbucket, GitHub, and HipChat and how they are used for processes such as collaboration and code sharing.

**Target Audience**

This course is designed for software professionals interested in learning DevOps or implementing DevOps in their organization

**Expected Duration**

1 Hours, 56 minutes

**Prerequisites**

None

**objectives**

**Course Introduction**

**The First Step in the DevOps Methodology**

 describe the DevOps structure and methodology

**Brief History of the Traditional SDLC**

 recognize the patterns and evolution of a traditional SDLC, and how DevOps grew out of it

**Problems Solved by Using DevOps**

 compare the traditional SDLC with DevOps and recognize how DevOps is used to solve software development problems

**Rethinking the SDLC with DevOps**

 describe how DevOps can be used to replace the traditional SDLC

**Factors Driving DevOps Acceptance**

 identify the factors involved in the widespread acceptance of the DevOps methodology

**New Challenges of the DevOps Methodology**

 name the challenges created by the adoption of the DevOps methodology

**DevOps Acceptance and Usage**

 list the major users of DevOps and describe reasons for its acceptance and adoption

**Hardware Provisioning**

 describe how DevOps is used to replace traditional hardware provisioning tasks

**Configuration Management**

 compare traditional configuration tasks with DevOps and recognize DevOps configuration tools such as Chef and Puppet

**Creating Storage and Databases**

 use DevOps to create back ends for your applications

**Providing Security**

 configure application security through DevOps

**Virtualization**

 provision virtual servers through DevOps and recognize DevOps virtualization tools such as Vagrant

**Operations**

 compare traditional operations tasks with DevOps

**Introduction to the DevOps Workflow**

 describe the DevOps workflow

**Requirement Gathering with DevOps**

 identify how DevOps is used when gathering software requirements

**The DevOps Development Cycle**

 recognize how DevOps changes the way software is developed

**QA and User Acceptance Testing**

 distinguish between the DevOps stages of quality assurance and user acceptance

**Application Deployment Using DevOps**

 perform application builds and deployments using the DevOps methodology

**Using DevOps for Maintenance and Release Scheduling**

 define the steps for DevOps software releases and maintenance scheduling and recognize DevOps release tools such as Jenkins

**Using HipChat for Distributed Team Management**

 describe how HipChat is used to manage geographically separated teams

**Using GitHub for Collaboration**

 specify how collaboration occurs with GitHub

**Sharing Software Issues with JIRA**

 describe how JIRA is used to log and share software issues

**Aligning Teams Using Confluence**

 use Confluence for parallel team management

**Sharing Code with Bitbucket**

 use Bitbucket for code sharing and versioning

**Managing Cross-Platform Development with DevOps**

 describe how DevOps is used to manage cross-platform development issues

**Exercise: Set up DevOps Processes and Tools**

**Course Number os\_devo\_a01\_it\_enus**

DevOps Fundamentals: Tools, Technologies, and Infrastructures  
Course Objectives

**Lesson: Overview, Definitions, and Terminology**

After completing this topic, you should be able to

* start the course
* describe the DevOps structure and methodology
* recognize the patterns and evolution of a traditional SDLC, and how DevOps grew out of it
* compare the traditional SDLC with DevOps and recognize how DevOps is used to solve software development problems
* describe how DevOps can be used to replace the traditional SDLC
* identify the factors involved in the widespread acceptance of the DevOps methodology
* name the challenges created by the adoption of the DevOps methodology
* list the major users of DevOps and describe reasons for its acceptance and adoption

**Lesson: Comparison to Traditional SysAdmin Roles**

After completing this topic, you should be able to

* describe how DevOps is used to replace traditional hardware provisioning tasks
* compare traditional configuration tasks with DevOps and recognize DevOps configuration tools such as Chef and Puppet
* use DevOps to create back ends for your applications
* configure application security through DevOps
* provision virtual servers through DevOps and recognize DevOps virtualization tools such as Vagrant
* compare traditional operations tasks with DevOps

**Lesson: Processes**

After completing this topic, you should be able to

* describe the DevOps workflow
* identify how DevOps is used when gathering software requirements
* recognize how DevOps changes the way software is developed
* distinguish between the DevOps stages of quality assurance and user acceptance
* perform application builds and deployments using the DevOps methodology
* define the steps for DevOps software releases and maintenance scheduling and recognize DevOps release tools such as Jenkins

**Lesson: DevOps Technology: Functions and Groupings**

After completing this topic, you should be able to

* describe how HipChat is used to manage geographically separated teams
* specify how collaboration occurs with GitHub
* describe how JIRA is used to log and share software issues
* use Confluence for parallel team management
* use Bitbucket for code sharing and versioning
* describe how DevOps is used to manage cross-platform development issues

**Lesson: Practice: Setting up a DevOps Environment**

After completing this topic, you should be able to

* describe the software development life cycle within an organization and be able to recommend DevOps processes and tools

# **Course Transcript**

# DevOps Fundamentals: Tools, Technologies, and Infrastructures

## **Overview, Definitions, and Terminology**

| [1. Course Introduction](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#t2) |

| [2. The First Step in the DevOps Methodology](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#t6) |

| [3. Brief History of the Traditional SDLC](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#t10) |

| [4. Problems Solved by Using DevOps](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#t14) |

| [5. Rethinking the SDLC with DevOps](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#t18) |

| [6. Factors Driving DevOps Acceptance](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#t22) |

| [7. New Challenges of the DevOps Methodology](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#t26) |

| [8. DevOps Acceptance and Usage](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#t30) |

## **Comparison to Traditional SysAdmin Roles**

| [1. Hardware Provisioning](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#t35) |

| [2. Configuration Management](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#t39) |

| [3. Creating Storage and Databases](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#t43) |

| [4. Providing Security](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#t47) |

| [5. Virtualization](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#t51) |

| [6. Operations](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#t55) |

## **Processes**

| [1. Introduction to the DevOps Workflow](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#t60) |

| [2. Requirement Gathering with DevOps](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#t64) |

| [3. The DevOps Development Cycle](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#t68) |

| [4. QA and User Acceptance Testing](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#t72) |

| [5. Application Deployment Using DevOps](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#t76) |

| [6. Using DevOps for Maintenance and Release Scheduling](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#t80) |

## **DevOps Technology: Functions and Groupings**

| [1. Using HipChat for Distributed Team Management](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#t85) |

| [2. Using GitHub for Collaboration](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#t89) |

| [3. Sharing Software Issues with JIRA](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#t93) |

| [4. Aligning Teams Using Confluence](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#t97) |

| [5. Sharing Code with Bitbucket](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#t101) |

| [6. Managing Cross-Platform Development with DevOps](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#t105) |

## **Practice: Setting up a DevOps Environment**

| [1. Exercise: Set up DevOps Processes and Tools](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#t110) |

# Course Introduction

## Learning Objective

After completing this topic, you should be able to

* *start the course*

## **1. Introduction to the course**

Welcome to DevOps! DevOps is a software methodology where the operations and development staff participate together in the entire software lifecycle, from the design phase through the development and production phases. In this course, you will learn the fundamentals of the DevOps methodology, and the principles and processes of the DevOps workflow. You will also be introduced to various DevOps tools such as JIRA, Confluence, Bitbucket, GitHub, and HipChat and how they are used for processes such as collaboration and code sharing.

[Back to top](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#top)

# The First Step in the DevOps Methodology

## Learning Objective

After completing this topic, you should be able to

* *describe the DevOps structure and methodology*

## **1. Structure and methodology of DevOps**

DevOps is a software development methodology that stresses intercommunication between software developers and other personnel such as testers, designers, quality assurance, and operations**.** Since the term was initially coined to describe the collaboration between software developers and operations personnel, both roles were concatenated to form the word DevOps. DevOps is a direct descendant of the more established Agile software methodology. DevOps is fairly new. The term was first used at an Agile conference in 2008. The name has stuck and has created a full-blown methodology. DevOps has created new ideas about how organizations create and maintain software. DevOps has also created an ecosystem full of tools and other solutions that claim to be DevOps. The goal of DevOps is to leverage collaboration throughout the entire software development life cycle. Another cornerstone of DevOps is blurring some of the traditional responsibilities of the personnel involved in software development. DevOps specifies the certain responsibilities and tasks that traditionally would be held by a single group – that is developers write test code and operations personnel set up servers – are now shared. This collaboration will result in faster development times, better quality code, increased efficiency, quality assurance, user acceptance, and production support. Of course, since DevOps is so new, there are still many questions on how to measure the benefits of DevOps.   
  
There are many underlying reasons that can be attributed to the emergence of DevOps. A major factor is the evolution of software. Traditionally, software was developed to complete a business or technical task and programmers wrote them. This software program needs to run on a computer. Network operators would provision and maintain the computer on which developers program their software. Generally, there are two different groups here – developers and operations personnel. Each group has a mutually exclusive, predefined role. Recently, software has evolved to perform tasks such as emulation and virtualization. Developers can now deploy programs on virtual machines, often without the assistance of operations provisioning a physical server. Clearly, there needs to be collaboration and coordination between developers and operations. This collaboration is called DevOps. Of course, DevOps is more than this. DevOps collaboration is not limited to developers and operations. Adopters of DevOps identify different groups within their software infrastructure and the tasks they perform. Each group and task is analyzed to identify any areas of interdependence that can be added as collaborative efforts. DevOps intends to blur the traditional lines within the software development infrastructure. It is hoped that – since traditional tasks are now a shared collaborative effort – each group will feel more invested in tasks that they did not traditionally perform. To name a couple, developers will write better code as they are now partners in the quality assurance process and operations staff may spend more time on security patches as they discover network impact on the software development life cycle.   
  
So what does all this mean? Do developers now get paged when a server goes down? Are operations personnel included in software requirement meetings? Do both groups join quality assurance team to create a test plan for a new application? In many cases, the answer is yes. While not relinquishing their traditional roles, each group is invested in different portions of the software development life cycle – portions that they were not involved with until now. Adopters of DevOps claim that this collaboration alone has a positive impact on software personnel. Since all personnel is involved in the software development life cycle, they take more ownership of the tasks they perform and understand how their role impacts other teams and the organization as a whole.

[Back to top](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#top)

# Brief History of the Traditional SDLC

## Learning Objective

After completing this topic, you should be able to

* *recognize the patterns and evolution of a traditional SDLC, and how DevOps grew out of it*

## **1. SDLC and DevOps**

To understand DevOps, it helps to review the role of the traditional software development cycle or the SDLC. Over the years, there have been advances in software and hardware design. Decades ago, third-generation programs were developed, debugged, and deployed on lumbering reliable mainframes. These programs for the most part were batch processes performing mundane task, such as computing interest rates in bank accounts or updating customer purchase records. Slowly this software system evolved the fourth-generation languages running on commoditized servers. The next stage of evolution was for the software to run over the web or cloud over servers load balanced over the entire internet. Although there is a clear evolution of software systems, the process in which it was developed remained the same. After a period where requirements were gathered, the program development stage begins. In this stage, programmers use whatever language is at their disposal to design and program a piece of software that hopefully does something useful. Developers work together to work as a team performing programming and testing tasks. The development group has clearly defined roles. They write and test code. That's all they do. Before they write a program, they get the functional specifications what the program is supposed to do from a business analyst. After the program is written, it is given to a quality assurance group.   
  
The quality assurance group – QA, test the piece of software to insure it is stable and meets the requirements that were given to the developers. Quality assurance testers usually know little about software development. They also have little interest or knowledge of hardware. This group exhaustively puts the code through its paces either trying to break it or expose any shortcomings it might have. Any issues real or imagined are passed back to the developers to fix, then are passed back to QA to be retested. This process goes on for a predefined period until all issues have been addressed. Developers and quality assurance testers have predefined roles. These roles rarely overlap. Next the tested code makes its way to the operations group responsible for installing the application. The application is placed on a computer somewhere and presumably it works and all is right with the world. The operations group is responsible for providing the computer and the network. They are also usually responsible for maintaining the operating system of the computers within the organization. They know little or nothing about software development and have no interest in testing. They are concerned with provisioning new servers, updating hardware, and maintaining operating systems with the current patches et cetera. The role of the operations personnel usually does not overlap into development or testing tasks.   
  
This traditional software development cycle works well in a predefined highly rigid environment. Back in the day, computer languages were much simpler and would only run on one piece of vendor supplied proprietary hardware. The program may have been written in HP COBOL only to be run on an HP3000. The modern process is more complex. A program may be written in a language that can run on a multitude of different servers and operating systems. Code needs to be more thoroughly tested to account for different platforms. Developers need to have knowledge on what the host system will support. Operations may need to know which version of a language a program is developed in. There are simply too many variables for the development, quality assurance, and operations group to act independently. There has to be a degree of collaboration. That collaboration is called DevOps.

[Back to top](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#top)

# Problems Solved by Using DevOps

## Learning Objective

After completing this topic, you should be able to

* *compare the traditional SDLC with DevOps and recognize how DevOps is used to solve software development problems*

## **1. Traditional SDLC and DevOps comparison**

DevOps augments the traditional software development cycle, it does not replace it. By augmenting traditional software tasks with the collaboration that DevOps provides, problems that exist in a non-DevOps environment can be addressed. The main issue in a non-DevOps environment is the lack of collaboration. When groups operate in silos, they have little knowledge on how their work impacts other teams. Also teams working in a silo have little knowledge on what other teams actually do. This lack of collaboration has obvious drawbacks, such as the lack of flow of ideas and innovations. There are some other drawbacks on a noncollaborative environment that are more concrete and easy to identify. Collaboration between functional designers and developer reduce software development time. Using DevOps between these two groups will result in better understanding and what the software application should do. In a non-DevOps environment, the functional designers have requirement meetings with the users to determine what a software application should do. Developers take this design and code an application. The developers have no real idea on the value of the application has to the end user or the organization as a whole. The functional designer has little understanding on how an application is actually develop. Using DevOps the coders can attend design meetings and the functional designers can attend development meetings. Both teams learn about each other's role in the software development cycle.   
  
DevOps between developers and quality assurance testers increase software quality and reduce the testing time. In a traditional scenario, coders don't know how an application is to be tested, and testers have little idea of the features coded into an application. Collaboration here is more valuable than just sharing job knowledge. If the developers knew how the application would be tested, they may employ a testing tool to assist in finding bugs. In addition to collaboration, DevOps is about tools and technology. In a DevOps approach, developers collaborate with the quality assurance testers and determine the testing process, and which portions of the testing can be automated. In this fashion, the application has already been pretested when it's passed from development to quality assurance. DevOps between quality assurance and operations can prevent deployment showstoppers. After an application has passed the quality assurance stage, it is passed on the operations to be installed. Many organizations have different environments for application testing and application deployment. An application may be tested on a Windows 32-bit machine, but may be deployed on a Windows 64 box. This has some obvious risks. Using DevOps the quality assurance team can work with the operations group to ensure that the application is tested on the same platform it will be installed on, or perhaps on virtual machines that would duplicate the production environment. DevOps combines knowledge and technology to mitigate any platform issues.   
  
DevOps between development and operations improves the consistence of application behavior. Bugs may be reported on a production system that can't be reproduced by a developer on local machine. As applications have evolved, so have their dependencies. Applications are no longer standalone. Most are distributed and are heavily dependent on the platform and the operating system. Most web applications are also heavily dependent on plugins and even the version of the operating system and runtime environment. This complex mix of technology makes it difficult to debug issues. In some cases, the runtime environment is so different than the development machine bugs cannot be duplicated at all. DevOps helps with this by supplying tools that emulate production environments.

[Back to top](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#top)

# Rethinking the SDLC with DevOps

## Learning Objective

After completing this topic, you should be able to

* *describe how DevOps can be used to replace the traditional SDLC*

## **1. Replacing traditional SDLC with DevOps**

DevOps allows us to rethink the traditional SDLC. The traditional SDLC has been around for years and has relatively unchanged. The SDLC is divided into the following phases – requirement gathering, development, testing, deployment, and maintenance. Each of these stages is a silo. Silos could be identified and broken down. Each of these silos can be eliminated. Collaboration will be increased in a DevOps system as different groups learn to work together on a common goal. DevOps tools can be used to perform collaborative tasks. Teams can leverage tools that automate previously laborious tasks. The ultimate goal of DevOps is to improve quality. With traditional barriers removed, the quality of software applications increases. The software development cycle starts with requirement gathering. In a DevOps system, requirement gathering is not the first phase but the phase before deployment, after deployment, and after maintenance. Traditionally this phase is closed door and only the initial stakeholders are involved. Stakeholders from other groups, like development, are usually not invited. DevOps tools and techniques allow more personnel to be involved. With collaborative tools, more groups are involved with the gathering of requirements. Newly developed software is viewed as an enterprise-wide venture and not an IT project. Now, that all stakeholders can see outside their silo, their groups can collectively see the impact of the application across the business.   
  
Applications then enter a development phase of software development. This is where applications get written. Here groups of developers write code. Developers traditionally work in a black box. In most cases, the business or other groups will not see the programmed application until it leaves development. Other personnel within an organization have little input in the development process. It is up to the developers to design and write the code. DevOps opens up the development phase. In a DevOps environment, members of other groups are allowed to participate. DevOps allows the development of software to occur in a white box. In a white-box environment, the business has a look into the entire development process. DevOps reinvents the quality assurance phase of the software development life cycle. This is the phase that occurs after development. It's where software is tested. Quality assurance is now a continuous process and no longer a siloed phase. In a non-DevOps environment, quality assurance only occurs here. Quality assurance is now a continuous process. Members from this group can now work through the entire SDLC. DevOps tools and techniques are used to test software quality throughout the entire QA life cycle. These tools can help developers and testers work on test plans. QA testers are now sourced and embedded in other software groups.   
  
DevOps changes the way applications are built and deployed. In the traditional SDLC, building and deploying applications was a cumbersome error-prone task. DevOps tools allow for the unprecedented automation of software staging, promotion, and deployment. With automation, building and deploying become more reliable. DevOps tools are used to monitor the performance and the health of production applications. Feedback in this phase can be used to drive new application requirements. The entire organization is now part of the application deployment process. In DevOps, even developers can deploy applications. DevOps tools are also used for bug tracking and release management. These tools help the organization approve the overall quality of the application.

[Back to top](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#top)

# Factors Driving DevOps Acceptance

## Learning Objective

After completing this topic, you should be able to

* *identify the factors involved in the widespread acceptance of the DevOps methodology*

## **1. Acceptance of DevOps methodology**

The DevOps methodology is relatively new and emerging, but adoption has been rapid. DevOps has only been around since 2008, but it's already been adopted by major organizations. There is somewhat of a disagreement on how to measure the benefits of using DevOps. Most of the benefits are intangible and difficult to measure. It is generally accepted to be beneficial to organizations that have accepted it. DevOps improves quality while reducing the cycle time of the software application. There are many factors in determining DevOps acceptance. Technical skillset as well as system architecture are common considerations. Factors are not always financial. As a matter of fact, factors are usually quality based. But improved quality will drive down costs in a long run. The factors driving DevOps are many and are unique to each organization. Just as every organization is unique, so are the reasons for adopting DevOps. Most reasons have to do with increased quality. Other reasons have to do with how fast something can get done. DevOps tools build in efficiencies, rather than performing traditional manual task such as server configuration, new tools automate processes. DevOps applications are developed faster because of the improved quality and increased efficiency. Hardware costs can be lowered through the use of virtualization. Virtual machines are slowly taking over and replacing physical provisioning. They are easy to work with and are cheaper. Automated deployment takes fewer personnel. With fewer manual processes and a reduction of deployment errors, applications can now be deployed by fewer people.   
Heading: Factors Driving DevOps Acceptance.A pie chart showing the DevOps adoption percentage among organizations is shown. The pie chart shows that 16% of the organizations do not know what DevOps is, 39% of the organizations have already adopted DevOps, 27% of the organizations plan to adopt DevOps, and 18% of the organizations have no plans to adopt DevOps.   
  
Software quality is also a factor in driving acceptance. Quality software does not break, thus not only is it cheaper to develop, but it's inexpensive to maintain. DevOps applications have less scope creep. Because the application is designed, developed, and deployed by the entire organization, fewer added features are likely to be added. Applications spend less time in the formal testing phase of the software development life cycle. Because quality is now a culture and no longer a phase, the time an application spends in QA is greatly reduced. Higher quality software results in lower downtime. Applications that don't break, don't need expensive developers to fix them. Higher quality DevOps applications are more stable when deployed to production. Quality is ensured throughout the entire SDLC. Production and maintenance phases are usually pretty stable in the DevOps shop. DevOps builds a cohesive software development team. With silos removed, teams now overlap effectively forming one unified team. Applications are aligned to meet organizational goals. Before DevOps, it was possible for applications to be written that only served small portions of an organization. Applications are seen as meeting business objectives rather than serving smaller needs of a department. All stakeholders are sourced and are responsible for the success of an application. DevOps eliminates the "not my problem" syndrome by involving all team members throughout the entire SDLC.   
Heading: Factors Driving DevOps Acceptance (Continued).A pie chart showing the factors driving the demand for DevOps is displayed. In the pie chart, 27% is due to the greater need for simultaneous deployment across different platforms, 42% is due to improved quality and performance of the application, and 31% is due to improved end customer experience.   
  
DevOps standardizes the deployment process. Application deployments are filled with drama. There are just too many manual steps. DevOps build and deployment tools can be configured to build in redundancy. This built-in redundancy removed many of the errors that resulted in manual processes. All applications can now have the same build and deploy process. Standardizing the build and deploy process will make the process faster. The deployment process can now have fewer steps. Built-in redundancy builds solid repeatable deployment plans. Deployment and release management can easily be integrated into bug reporting. Deployment is not the end of the cycle for an application, but really just the start of a new requirement-gathering phase.

[Back to top](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#top)

# New Challenges of the DevOps Methodology

## Learning Objective

After completing this topic, you should be able to

* *name the challenges created by the adoption of the DevOps methodology*

## **1. Challenges of adopting DevOps**

There are challenges to usage and acceptance. DevOps certainly has created a culture clash. Not all organizations have bought in to the DevOps methodology. Even adopters of DevOps are somewhat skeptical of real and perceived benefits. DevOps is perceived differently by different groups and organizations. There is no book on DevOps. It's a methodology not a tool. Some DevOps processes have been around for a while and DevOps may just rebrand something an organization is been doing for a while. There is no universal agreement on if DevOps has any benefits at all. Some say that DevOps costs more in the long run. Benefits – if any – are esoteric, nontangible, and difficult to measure. There is no real dollar amount that can be subtracted or added by adopting DevOps practices. DevOps adds complexity to the software development life cycle. By moving development out of its siloed phase, the interaction between groups becomes complex. DevOps is another process that must be monitored and managed. These additional management tasks must be assigned to a manager, thus taking up more of his or her time. DevOps has a cost-benefit implementation curve. In the early implementation stages, DevOps will add significant time and confusion to the SDLC. Only after an initial shake-out process will DevOps reap any potential benefits. DevOps adds redundancy that may not be needed. In the traditional SDLC, QA testing occurs in its own phase. DevOps puts QA testers elsewhere. DevOps may take simple matters and make them more difficult than needed. If you have a small shop, does it make sense to adopt DevOps processes?   
  
Functional groups may see DevOps as a threat. Groups are used to work in a specific way. Change can be perceived bad to many people. If it's not broken, why fix it? Many shops running a waterfall approach of the SDLC see DevOps as yet another methodology destined to fail. They will point to the fact that any benefits of DevOps are intangible. DevOps may upset the software development corporate culture. DevOps is about complete culture change, not about people change. For example, an organization may have to alter or even eliminate their relationship with physical server providers. DevOps needs universal acceptance to work. There is no partial adoption of DevOps. It's really an all or nothing deal. DevOps may be implemented differently by different groups within an organization. An organization may have multiple IT departments, each with its own ideas and implementation of DevOps. DevOps tools vary greatly in features and quality. Some are created by large vendors and are high quality; some are shareware and pretty low quality. Some of the tools are platform dependent or require complex configuration. Learning and using new tools may be more complex than a manual process they replace. Robust tools become very complex and can take a while to learn. There is a real possibility that learning a new tool may be more difficult than a task that needs to be completed. There is no agreement on what even constitutes a DevOps tool. Some tools predate the DevOps moniker, but they are DevOps tools nonetheless. The DevOps ecosystem is unorganized and hard to navigate. There are no real standards. DevOps tools and processes have "I know it when I see it" kind of property. Not everyone agrees on the formal definition of DevOps, but we can all spot a DevOps tool or process.   
  
DevOps requires a financial commitment. Nothing in the world is free and neither is DevOps. Tools must be evaluated and implemented. Even free tools require nonfree people to test and evaluate them. Most enterprise tools are expensive. Most freeware tools are just gateways into for-sale enterprise versions. There is almost always a catch in using free tools. Companies don't make money-giving things away. If you like a free version of a tool, there is almost always enterprise version for sale. Personnel may need formal training in DevOps processes and tools. Some tools such as Puppet or Vagrant will not be learned on the fly. Training is needed and it can be expensive. There is no clear answer to the return on investment, if any, in adopting DevOps. Critics often point out that implementation of DevOps won't be considered until the benefits of DevOps are more quantifiable.

[Back to top](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#top)

# DevOps Acceptance and Usage

## Learning Objective

After completing this topic, you should be able to

* *list the major users of DevOps and describe reasons for its acceptance and adoption*

## **1. Reasons for the acceptance of DevOps**

Amazon was an early adopter of DevOps. They were also the first online retailer to take on this new approach. Amazon uses DevOps to deploy applications over 300 times an hour. Amazon also deploys applications from anywhere in its pipeline. Through DevOps tools, Amazon has the capacity of over 100 deploys an hour. Because there is a high degree of confidence in the deployed software, Amazon considers it tested and ready for production. The deployment process is standard throughout the entire IT infrastructure. Standardization is achieved through DevOps pipeline in-built tools. Amazon deploys to over 10,000 servers. That number is according to them. Some insiders consider this number conservative. Netflix has used DevOps since 2013. The online movie rental company uses DevOps to build in efficiencies into their IT infrastructure. Netflix uses DevOps to manage over 100 releases a day. To achieve this, there are big use of automated configuration and virtualization tools. Application startup, configuration, and code deployment is handled through DevOps. Everything is handled the same way. Netflix attempts to automate everything. With automation comes predictability. Predictable processes run smoother and are less error prone. DevOps processes and tools are at the center of the Netflix software development life cycle. Being a fairly new company, implementing DevOps was much easier as there's not a long-standing IT culture.   
  
Etsy uses DevOps to manage over $1 billion in transactions every year. DevOps helps them standardize processes and commoditize infrastructure. They have over 200 code committers, everyone is expected to deploy. The IT staff at Etsy is not even divided into developers and operations. Everyone does about the same thing. Engineers are expected to deploy on the first day of the job. Etsy allows developers to deploy code on the fly using build and deploy tools. Etsy points to DevOps for creating a "Software as a culture" development environment. It helps them create and deploy their own brand in a highly competitive marketplace. Flickr handles 3 billion photos – 4,000 photos per second. Flickr is also a pioneer in Big Data and Big Data tools. Even though Flickr embraces technology, it uses it to eliminate as many IT processes as possible. Flicker claims that DevOps enables the business by all, but eliminating traditional IT tasks. Flickr is an extensive user of Enterprise DevOps tools. These tools help manage the large volume of transactions that occur on their servers. DevOps tools allow Flickr to concentrate more on core business issues. As they say, they are not an IT shop.   
  
WebMD is an online medical source. DevOps helps them continually publish time-sensitive content. DevOps fosters continuous delivery and feedback. WebMD is more of a content provider than a web application organization. Customer feedback drives new content pushes. DevOps reduces overhead with push button deployment. DevOps also quantifies change and stability and offers transparency for software and process compliance. At WebMD, deployment was reduced from 2 days down to 60 seconds.

[Back to top](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#top)

# Hardware Provisioning

## Learning Objective

After completing this topic, you should be able to

* *describe how DevOps is used to replace traditional hardware provisioning tasks*

## **1. DevOps and hardware provisioning**

Hardware provisioning used to be a "do it yourself" task. Most organizations had someone in-house that would be able to put a computer together. Hardware was kept locally, sometimes very locally, like in the closet. The term of "hosting" only came from the need of outsourcing knowledge or the need for a small company to ramp up quickly. Large organizations had to have hardware expertise. Large networks were complex and did not lend itself to hosting. Maintaining your own network is expensive, and organizations were keen to offload this responsibility whenever a solution was available. Application development was considered a problem of IT, and applications often were hosted on cumbersome physical servers. Data centers were the next evolutionary step for hardware provisioning. Data centers would administer most hardware and provisional tasks. Data centers essentially moved hardware and did nothing to reinvent the process. This effectively took operations out of the organization and placed it in the outsourced data center. This did not completely put operations out of a job. Organizations often had to maintain operations staff to deal with engineers at the data center. Data centers often added complexity to hardware provisioning.   
  
Managed hosting was the next evolutionary step in hardware provisioning. Managed hosting basically outsourced network operations from the data center to the managed hosting company. The managed hosting company now took over all operational portions of the SDLC. Organizations can now focus on the "Dev" portion of software development. Many organizations liked this as it got them out of the hardware provisioning business. Engineering personnel – "Ops" were managed by the hosting company. Separation between Dev and Ops often led to problems. Now that these two processes are geographically separated, there may be communication problems between the two groups. Cloud provisioning is the current evolutionary step in hardware provisioning. The term hardware is used loosely here as servers are virtual on the cloud. Albeit, even virtual servers have to run on some physical server. Cloud provisioning reunites developers – Dev – with operations personnel – Ops – as the infrastructure can now be managed as code. Operations often can perform provisioning by a few mouse clicks, even developers can perform virtual provisioning. Cloud provisioning has further blurred the line between Dev and Ops.   
  
With the cloud, hardware provisioning becomes less about hardware. High capacity throughput servers are used to host hundreds or more virtual environments. Most servers are no longer physical. Many organizations that traditionally have physically provisioned hardware are adopting virtualization. The provisioning task is becoming more encapsulated and automated as the use of virtual servers increases. The provisioning process is a lot like managing software. Provisioned code can even be versioned and source controlled. Dev and Ops continue to converge as both roles continue to overlap.

[Back to top](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#top)

# Configuration Management

## Learning Objective

After completing this topic, you should be able to

* *compare traditional configuration tasks with DevOps and recognize DevOps configuration tools such as Chef and Puppet*

## **1. Different DevOps configuration tools**

Configuration management has traditionally been a manual task. In physical or virtual provisioning, configuration task can be troublesome if they are nonautomated. Developers need servers in which to develop, test, and deploy their software. Configuration management usually falls under the Ops personnel of an organization. Traditionally, it is their job to create and maintain the infrastructure in which the organization conducts its business. Configuration management involves both hardware and software. Often the two are incompatible with each other. The increasing complexity of applications has made configuration management more troublesome. Complex applications can be buggy or not work at all on machines that they were not developed on. Configuration management has become more automated with DevOps. Manual steps have been identified and turned into configurable, repeatable processes.   
  
All phases of the SDLC have potential configuration problems. Software applications are traditionally passed and propagated through different software environments. Applications are often developed and tested on different platforms. Applications are often written in languages that have many external dependencies. Machine architecture and operating systems may also be different. Varying configuration makes bugs hard to track. Applications can exhibit very different behavior on different machines. Testing configuration can be more complex than testing the application itself. Varying configuration leads to code and testing instability. Configuration issues lead to "It works on my machine" syndrome, effectively telling operations that badly acting applications are not the problem of the developers. Operating system management is also a configuration issue. Many organizations run different operating systems on their hardware. Applications may be developed on a Windows desktop and deployed on a Linux server. This can further make an application tough to test. Bugs may be related to the server's operating system and not the application. Even servers from the same OS may have different version or updates. Windows is famous for small idiosyncrasies between versions and patches. Configuration management on Linux systems can be especially finicky. Different distributions have different behavior. Deployment issues grow exponentially as the number of deployment servers grows. Larger networks create larger problems.   
  
Applications have become more dependent on shared pieces of code. Application development is more about coding small applications in multiple languages. Gone are the days of fully-contained languages. Most applications use shared resources. Java and .NET applications rely on shared resources – JARs and Assemblies. These shared resources can vary in functionality and may even be buggy. Versioning becomes very important. Inconsistencies in plugins can also cause configuration issues. The same plugin may be available from different vendors and behave differently. Managing plugins, shared code, and other dependencies have become untenable. DevOps configuration management attempts to fix this. DevOps has processes and tools for automated configuration management. These tools standardize the configuration process, making it predictable and less error prone. Puppet and Chef are DevOps tools that automate configuration management. Both are considered competing products and each has its own features and challenges. Automated configuration tools are used to standardize the configurations of thousands of machines. In DevOps, there are no manual configuration processes. DevOps has automated and simplified configuration tasks. And since configuration scripts are code, DevOps also allows configuration management to be managed like software.

[Back to top](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#top)

# Creating Storage and Databases

## Learning Objective

After completing this topic, you should be able to

* *use DevOps to create back ends for your applications*

## **1. Application backup using DevOps**

In this demo, I'm going to show you how you can use your back-end databases as data sources for any kind of DevOps tools. Now, when you're using DevOps tools, you're going to use all kinds of databases. And these databases can be anywhere. You're going to use cloud databases, such as Cassandra or Redis. Or you're going to use enterprise databases, traditional legacy systems – such as Oracle, SQL Server, DB2, or Informix. It really runs the gamut of what kind of databases you're actually going to use. Now, when you use legacy databases – such as Oracle, SQL Server, or DB2 – there's a variety of different ways that you can actually connect to them from your DevOps tools. What I'm going to show you is I'm going to show you how you can use ODBC on the server to actually create a data source so you can connect your DevOps tools to that database.   
The Windows 8 Start screen is shown.   
  
Now I'm in Windows Server. I'm going to navigate down to Apps, navigate to the right to **PC settings**. From here I'll select the **Control Panel**. And this should open up my Control Panel. I'll navigate to **System and Security**. From here I'll navigate down to **Administrative Tools**. And, on your right-hand side, you should see two icons for creating ODBC data sources. Now you're going to create either 32-bit or 64-bit data source depending on the version of the database that you have. A 64-bit database will require a 64-bit ODBC source. And a 32-bit database would require a 32-bit data source. So I'm going to click **ODBC Data Sources (32-bit)**. This will open up the ODBC Data Source Administrator (32-bit) window where you get to see all your existing data sources for either a user or for all users. I want to click **System DSN**. And, as you see here, I don't have any data sources actually configured.   
The presenter navigates to the Apps folder and clicks the PC settings icon to open the PC settings page. In the PC settings page, he clicks the Control Panel link to open the Control Panel window. Then he clicks the System and Security link to open the System and Security window. He then clicks the Administrative Tools link to open the Administrative Tools window. In the Administrative Tools window, the presenter points to two entries, ODBC Data Sources (32-bit) and ODBC Data Sources (64-bit). The presenter clicks the ODBC Data Sources (32-bit) entry to open the ODBC Data Source Administrator (32-bit) dialog box. The dialog box has several tabs such as User DSN, System DSN, and File DSN, among others. The presenter clicks the System DSN tab to open its tabbed page.   
  
So to do that I would click **Add**. And, in this list, you're going to see all of the ODBC data sources that are installed on this local machine. Now the list is going to differ from machine to machine because each machine may have different ODBC data sources or ODBC drivers actually installed on them. So, if you don't see your driver here for the database that you want to use...for example, if we're using Informix or DB2, we don't see an ODBC data source for that. You can go to that vendor website and download one. They're usually free, and they're usually very, very easy to install. So here we got data sources for two large legacy database management systems. We have an ODBC data source for Oracle and an ODB data source for SQL Server. Now, to set this up, you simply would give it a double-click. And I'm not going to go over each specific step here because they're going to be different depending on the database management system that you're using. But most of them you put in the Name. Now I'm going to put in DevOps Example. And, for the Description, I might do the same. And each of the installs have a place where you specify what server or which database are you going to connect to. Now again, this is going to be different depending on the database you're using. So I'll skip that step because it's going to be very dependent on the specific driver. But anyway, once you set up your ODBC data source, you can connect that ODBC data source from your DevOps tool.   
The System DSN tabbed page is open. It has the System Data Sources section. The section has a blank table with three headers: Name, Platform, and Driver. Next to the section are the Add, Remove, and Configure buttons. The presenter clicks the Add button to open the "Create New Data Source" dialog box. The dialog box has the "Select a driver for which you want to set up a data source" section. The section contains several ODBC data sources. The presenter points to two data sources, Microsoft ODBC for Oracle and SQL Server. The presenter double-clicks the SQL Server data source and the "Create a New Source to SQL Server" dialog box appears. The dialog box has three text fields: Name, Description, and Server. The presenter adds the text "DevOps Example" in the Name and Description text fields.

[Back to top](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#top)

# Providing Security

## Learning Objective

After completing this topic, you should be able to

* *configure application security through DevOps*

## **1. Application security using DevOps**

Application security is often afterthought of a DevOps plan. Security is part of DevOps. Its role is somewhat convoluted as security does not cleanly fall into 'Dev' or the 'Ops' group. Application security does not have a discreet stage in the SDLC. It's, sort of, integrated everywhere. Security administrators and DevOps personnel work to integrate application security. Adding security into a system often occurs while the application is being coded. CISOs and security teams have a RAD approach to operational security. Goals in software development and implementing security can be very different. Often security goals are short term like to pass an audit. Achieving short-term goals is not the main objective of a DevOps system. DevOps is more strategic and security is more tactical.   
  
DevOps and application security can be at odds with each other. DevOps attempts to streamline a process while security attempts to methodically trudge through it. Integrating security into a DevOps plan can be challenging. Like quality, security should follow DevOps applications throughout the entire pipeline. There is no phase for security. Security must be built into the system and not be an afterthought. The business only invests in things it understands. If the business invests in DevOps because it produces inexpensive software, they may not understand why a slower process, such as, integrating security should be DevOps. Other than passing short-term audits, many organizations do not invest much in security. There is a positive correlation between DevOps decreased cycle time and the bottom line. Any increased time it takes to properly integrate security will not offset DevOps savings. This correlation makes the decision to invest in DevOps pretty simple. Integrating security into DevOps may slow the process down, but that's not a bad thing. Organizations need to be educated about the value of DevOps integrated security. In DevOps, you can still have fast, good, inexpensive, and secure.   
  
DevOps means more than DevOps and R&D. It also means security. Security has become a culture in DevOps applications and systems. Security should be included as early as possible in a DevOps application. All stakeholders are responsible for the security application and the organization as a whole. The goal is not to uncover security problems but to prevent them. DevOps is about security too, not just faster deployments. Security personnel need to be included in selecting DevOps tools. Security personnel is part of the DevOps team and will use tools to work with other groups. DevOps provides a huge opportunity to align security needs to the needs of the business. Application security is often implemented on the application level and not at the enterprise level. DevOps allows organizations to view security as a means to protect a business not just applications. Security controls are implemented earlier in DevOps applications. Security is now part of the requirement-gathering phase. DevOps allows security to be baked in with the SDLC. No longer security in afterthought. Other groups learn more about security and how it impacts their applications and the business as a whole. Security through DevOps adds value to the SDLC. Secure applications are more inexpensive than nonsecure ones.

[Back to top](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#top)

# Virtualization

## Learning Objective

After completing this topic, you should be able to

* *provision virtual servers through DevOps and recognize DevOps virtualization tools such as Vagrant*

## **1. Virtual servers and virtualization tools**

Virtualization is replacing physical machine provisioning. More and more organizations are embracing cloud technologies. Physical provisioning is slow, error prone, and expensive. Physical machines are also difficult to maintain and, quite frankly, take up too much room. Virtual machines can be provisioned almost instantly. Gone are the days of shopping for servers or awaiting the deliveryman. Server provisioning has usually been an operations task. Instant virtual provisioning has allowed developers to provision their own servers. Provisioning is no longer solely an operations task. Virtualization makes server provisioning pretty simple. Virtualization fits in nicely with the DevOps methodology. It takes a task once assigned to a single group and allows just about anyone to do it. Virtual servers can be created by anybody involved in the SDLC. QA testers have been known to spin up a VM for various testing scenarios. Instant virtualization has led to new cool technologies and tools. Most of these tools are still in their infancy. Platform-as-a-Service solutions view platforms as software. Entire virtual platforms can be created with a few lines of code. Tools like Vagrant allow automation of virtualization, spinning up an almost endless supply of virtual servers.   
  
Automation through Vagrant builds larger and more powerful infrastructures. Virtual environments can be created and modified. Vagrant can launch an unlimited number of virtual machines and provision them on virtual environments. Virtual machine automation is reliable and fast. This automation renders physical provisioning obsolete. Physical provisioning may soon be a thing of the past. Entire environments can be created and destroyed with a few mouse clicks. Vagrant opens up a world of possibilities as it allows us to rethink how we design, develop, test, and deploy applications. Vagrant standardizes runtime environments. As an application travels through the pipeline, it will live on exactly the same virtual box, configure the same identical way. Vagrant can be used to replicate identical physical servers. Configurations can be read on the physical machines then applied to its virtual counterpart. Software can be tested in only one environment. This eliminates issues when the testing platform may differ than the production one. Vagrant is easily configurable to allow fast on-the-fly changes if your network needs it. Configuration files are versionable. Like most DevOps tools, software runs the process and can be versioned in source control.   
  
Vagrant gives operations engineers disposable environments. Physical provisioning is not disposable. The physical architecture often drives what kind of application is designed and programed. In Vagrant, developers get consistent development and testing environments, as both virtual environments are configured to be the same. Vagrant allows the testing of "what if" scenarios like deploying applications and different operating systems. This makes testing more effective. Application stress testing is made more efficient through Vagrant through Vagrant's ability to simulate just about any virtual environment. Vagrant is flexible and works with most providers, such as VMware.

[Back to top](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#top)

# Operations

## Learning Objective

After completing this topic, you should be able to

* *compare traditional operations tasks with DevOps*

## **1. Comparing operations tasks with DevOps**

The DevOps methodology reinvents traditional operations. Before DevOps, operations personnel were organized by skill set. This has changed. Skill silos become dedicated cells. Traditional tasks such as scheduling deployments have changed as well. Scheduling becomes continuous and decentralized. It is no longer tied to the application promotion process. Information is created in smaller sizes and disseminated faster. Since personnel no longer works in silos, information is shared more freely rather than being stockpiled. Responsibilities are no longer being passed from one phase to another. DevOps changes the definition of "I'm done." Well in DevOps, you're really not ever done.   
  
DevOps reduces the risk for release management. Software releases are full of drama. They are also very slow and rigid. Many applications only have a few releases in its entire lifetime. This may be due to how error prone the release process actually is. Traditional release drama becomes a nonevent. In most cases, applications are almost always ready to be released. Applications are tested and promoted as part of the DevOps process. Errors are not sought out. In most cases, they are prevented. Any error or other issues are addressed as the application is being promoted. There is nothing really special about the production environment. Production is really just another environment promotion. DevOps manages failure better. Failure will happen in software development. The only way to prevent failure is by experiencing it. Traditional IT operations are built to avoid failure. In some ways, operations are in avert failure group instead of a group who performs operations tasks. Despite the incredible cost of failure avert measures, many projects do fail. And, when they do, they fail big and hard. DevOps does little to prevent failure. Failure is inevitable and should be managed if it can't be prevented. DevOps has a fail small, fail early, recover fast mentality. Managing small failures prevents large crash and burn scenarios.   
  
DevOps makes operations processes smaller. Small processes are easy to manage and measure. And they recover quickly from failure. Traditional shops use the waterfall methodology, which is big, slow, and many will say is hopelessly outdated. Over the years, operations tasks have consolidated. Large and cumbersome operation methodologies are disruptive. DevOps breaks large complex operations into smaller achievable tasks. Smaller tasks are easier to manage and to fix. DevOps changes the cost and capacity measurement model. This model has been used for software development operations for decades. Cost and capacity refers to what can get done and at what cost. It helps estimate the cost in dollars and resources for changes to the IT infrastructure – a cost-benefit estimator. DevOps adds a time-flow element to the cost and capacity measurement model. Flow looks at end-to-end cycle time and identifies areas of waste. By eliminating areas of waste, the software development process becomes cheaper. DevOps forces an organization to recognize the real cost of operational development and the cost of wasteful practices.

[Back to top](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#top)

# Introduction to the DevOps Workflow

## Learning Objective

After completing this topic, you should be able to

* *describe the DevOps workflow*

## **1. The DevOps workflow**

The DevOps workflow fundamentally changes the phases of software development. Many see this change as welcome as it refreshes the way we look at software development itself. The traditional phases of software development reflect a time in the past. IT teams in the past needed to be grouped by technical skill set. Traditional phases are large and encourage silos. Processes in these groups were engineered to avoid failure. Personnel were working in do not fail mode. Processes designed to prevent failure actually guarantee it. Loosening the noose from IT staff encourages the flow of ideas. Each of the phases of software development is made more agile through the DevOps workflow. The DevOps workflow aligns collective group focus. The whole team is now pulling for a common goal. Goals in a traditional system were never aligned. Often the goal was not to fail. The main objective was to promote the code or the project to the next group. And then the application becomes their problem. Different groups had different objectives. The only group, operations – whose main job was to successfully deploy the application – often no longer had the support of the other software groups. The real shared goal is to get a quality application into production.   
  
The DevOps workflow allows better team collaboration. Information is more plentiful and allows teams to share information and lessons learned. Traditional siloed teams need little collaboration. They collaborate it within their group, but almost never between groups. Collaboration was only needed for hand-off operations such as documenting what was done within that phase. Useful information often never left a group or a project phase. Improved collaboration creates an improved commitment-to-quality mind-set. A DevOps workflow allows for continuous improvement. Applications are often built and deployed in break-fix mode. Success is measured on a team's ability to fix bugs and redeploy the application. Traditional development focuses only on adding functionality to already buggy systems. Functionality is tested only until that works. And then no more testing takes place. DevOps encourages continuous improvement, not just testing functionality. Since applications are more stable in a DevOps environment, the focus is more on improving code – not just fixing it. Improved group collaboration and commitment leads to continuous process/software improvements.   
  
A DevOps workflow allows for continuous delivery. Applications are delivered all the time, every day, sometimes multiple times in a day. Contrast this with conventional delivery schedules, which may deploy an application every few months or even years. Traditional delivery systems are ridged, especially in their scheduling. DevOps is more flexible. Applications can exist and be deployed in all software phases simultaneously. As one application is deployed, the whole pipeline is advanced. Delivery is now continuous. Continuous delivery is a pillar of the DevOps methodology.

[Back to top](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#top)

# Requirement Gathering with DevOps

## Learning Objective

After completing this topic, you should be able to

* *identify how DevOps is used when gathering software requirements*

## **1. Gathering software requirements**

Most software projects begin with requirement gathering. Functional specifications are drawn up to explain the business process for which the application is being written. This is the software phase where it is determined what an application is supposed to do. The requirement-gathering process is usually insular. This phase can be very political and may even be secretive. In most cases, little attention is given to existing production systems and lessons learned from their deployment. New applications are often designed with wrong information and assumptions. DevOps greatly improves requirement-gathering phase by applying DevOps principles. DevOps allow software requirements to be based on user feedback as well as traditional functional design documents. Users can be internal business customers, not necessarily public users on the web. Before DevOps, requirements were usually written without substantial user feedback. In many cases, features were built into the application that were not needed or even asked for. In DevOps, requirement gathering is not the start of the development cycle, but just a stage in it. Feedback from maintenance and promotion feeds into the requirement-gathering phase. Requirements are compiled based on real-world monitoring and feedback and not on the feedback of the very few.   
Heading: Requirements Gathering with DevOps.The five software development stages in DevOps are requirement analysis, design, development, implementation and support, and maintenance and promotion.   
  
DevOps allows software requirements to reflect what is possible, not just what is asked for. Often, the capabilities of an organization are much greater than the software requirements ask for. DevOps uncovers the full capabilities of an organization and allows newly designed software to leverage these capabilities. Requirement gatherers are often nontechnical. DevOps adds technical staff to the requirement phase. Applications that don't exploit technology may be suggested. Requirements may not account for infrastructure issues. DevOps collaboration allows development and operations personnel to work with nontechnical requirement gatherers to build an application that exploits the full capabilities of an organization.   
  
DevOps allows software requirements to reflect current skill sets. Nontechnical skillsets are often overlooked in traditional waterfall-based systems where DevOps tends to leverage various skillsets across the enterprise. Requirements often do not exploit the organization's technical skillset. Programmers know the strong points of the languages they code in and operations personnel know how the network can be a corporate asset. DevOps allows emerging applications to exploit the capabilities of the technical staff. Requirements can also consider platform and support issues. DevOps allows requirements to reflect shared goals. Traditional requirement gathering may not reflect the goals of an organization. Different groups may write requirements that are not aligned. In DevOps, requirement gathering is performed by the entire business. DevOps applications are now built with requirements that reflect the vision of the entire organization.   
Heading: Requirements Gathering with DevOps (Continued).DevOps allows software requirements to reflect current skillsets. These skillsets are advanced analytics, business acumen, communication and collaboration, creativity, data integration, data visualization, software development, and system administration.

[Back to top](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#top)

# The DevOps Development Cycle

## Learning Objective

After completing this topic, you should be able to

* *recognize how DevOps changes the way software is developed*

## **1. DevOps software development cycle**

The development phase is where the application gets coded. Here functional requirements are translated into technical specifications. This phase is usually performed by a handful of developers. Most developers do not even see the functional requirement. To them, applications are nothing but code. Development is usually performed in a black box. Coders work with technical leads, coders may not even know what the application is supposed to do. Applications are often never seen by outside groups until completed. Insular development phase practices lead to other problems throughout the SDLC. DevOps allows for collaboration between developers and others within the organization. Other groups, although nontechnical, can offer insight on how the application is to be developed. Testers, for example, have a very structured approach to reviewing software. Software development lacks consistency and standards, other groups can suggest good practices. Also, developers are often geographically separated. DevOps has tools that help manage and communicate with remote teams. Different skillsets must be leveraged for a development staff. DevOps provides processes and tools to allow software teams to test and then share code.   
  
DevOps allows for collaboration between development and the business. It's the business's responsibility to expose the development staff to the reasons why an application is being developed. It is the responsibility of the developers to work with the business to show them what is possible. The business often does not see an application until it's in production. DevOps practices such as, having Joint Application Developer or JAD sessions allow collaboration between the teams. Interaction allows innovation and the sharing of ideas and developers understand the business purposes of the code. DevOps allows for collaboration between developers and quality assurance testers. Code quality becomes a shared goal with the ultimate goal being getting the application into production. DevOps processes and tools are used to automate testing in the development phase – almost nothing is manual anymore. Developers are exposed to the testing mindset and build quality into the code. Applications now spend less time in quality assurance and user acceptance testing. Lessons learned can now be returned to the requirement-gathering phase.   
  
DevOps allows developers to perform operations tasks. This is not a small point as this very concept is at the core of the DevOps methodology. It's curious as to why developers want to perform operations tasks while operations usually wants nothing to do with the developing software. Goals between development and operations can be very different. Developers want to write and deploy code and operations want to build and provision servers. In many cases, server and server configuration has very little to do with the applications that actually run on it. With DevOps, developers can build virtual networks to test "what if" scenarios. Operations can use DevOps to build virtual environments for software developers. Both groups are now in the provisioning business. DevOps manages expectations between development and operational personnel.

[Back to top](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#top)

# QA and User Acceptance Testing

## Learning Objective

After completing this topic, you should be able to

* *distinguish between the DevOps stages of quality assurance and user acceptance*

## **1. Stages of QA and user acceptance**

Formal software testing involves two phases. Each of the phases requires a different testing methodology. The first phase of testing is quality assurance testing or QA. QA testing is performed after the development phase. The quality assurance testers run through multiple levels of testing. Each attempting to find flaws in the software. System test cases are run to see if a code addition or change is working, as it should. Regression test cases run over portions of the applications that have not changed, just to make sure new bugs have not been introduced. User Acceptance Testing or UAT is performed after QA testing. In this phase, the user community runs the application, and ensures that it's up to standards of the group of people using it. In the traditional SDLC, there are now three levels of testing. Development applied unit test cases, and QA and UAT apply two additional levels of testing. DevOps streamlines this process and allows collaboration between QA and UAT.   
  
DevOps QA is about pre-empting defects, not finding them. Finding bugs is expensive. QA is not really a phase anymore. In many organizations, the QA groups still exist but their role has changed. QA personnel own the process of continuous improvement. At some level, QA personnel had become the true educators of DevOps. They are usually the first group to break their Silo and work with other teams in the organization. Their job is not only to test but find ways to make testing more efficient. Manual testing of software is all but eliminated. QA personnel are responsible for finding ways to automate testing. Automation is the key as complex manual testing is buggy in its own right. DevOps QA testers are the first to embrace quality as a culture. When this is achieved from upper management, down there are frontline staff, quality is infused in everything an individual does. This transformation improves every aspect of the organization, including the software. To help achieve this, DevOps QA testers are sourced to work with development and operations. Their job goes far beyond just finding bugs. QA testers have become business analysts and the champions of the quality process. DevOps duties include finding opportunities to improve processes and increase predictability.   
  
DevOps also changes the role of the UAT staff. Even though UAT staff performs application testing, they are not formal testers but members of the user community. Traditionally, UAT staff manually runs an application through business cases. If cases fail, the application goes back to the UAT phase. UAT staff now work with development to ensure business cases are accounted for in the code. DevOps uses UAT to ensure the application supports the business through all phases of the SDLC. DevOps all but combines QA and UAT. The user community teams up with QA. Both are advocates for continuous improvement in developing software applications. The QA and UAT phases still exist in the DevOps SDLC. Continuous improvement makes it everyone's job to design, build, and deploy quality applications. The testers' job is to prevent bugs and bad cases from getting into the QA/UAT phase. This is achieved by owning the quality process throughout the SDLC. Like other phases, DevOps shortens the traditional QA/UAT phase.

[Back to top](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#top)

# Application Deployment Using DevOps

## Learning Objective

After completing this topic, you should be able to

* *perform application builds and deployments using the DevOps methodology*

## **1. Application builds and deployments**

I'm going to give you a demonstration on continuous integration and the build and the deploy process. Now, before we go into the tool – which I have on the screen right here, I want to go over a brief overview of the build-deploy process as it exists today. Back in the old days, when we wanted to do a build what we would do is we would take all of our code, we would compile it, and then we would go through a process that would take the compiled code and make it into some kind of executable format – either an executable file or in the Java world we would build war files with jar files, and in .NET we do things with assemblies. And we have our own build process there. Where I'm getting at with this is the build process was usually very slow, it was very manual, and it was something that we only did once in a while – like maybe at the end of every week or at the end of every build-deploy cycle. What we would do is we would take all of our code, we would consolidate it, we would build it, and we would deploy it. Now DevOps looks at this a little bit differently. We have a concept now of continuous integration meaning that we do builds now, not maybe once a week or once every two weeks or once every build cycle, but we might do them every day, every hour, a few times an hour. How about we do a build every time there's code change? And when we do the builds now is we have an automated build tool that actually does that.   
The Jenkins home page is shown. The home page is divided into two areas. The area to the left includes five links: New Item, People, Build History, Manage Jenkins, and Credentials. The area to the right shows information about several Jenkins jobs.   
  
Now a few years back, maybe ten years back, there were some Java tools out there that would start to automate the build process, such as Ant. Ant comes to mind where you would build Ant scripts or maybe Maven scripts that would actually take the code and compile the code, put the jar dependencies that were in there, and then deploy the code. But Ant was, kind of, still kind of manual if you think about it, where there were still some steps in there that you had to kind of put in maybe some runtime parameters, et cetera. Now DevOps tools take that concept of automated builds to the nth degree, meaning that we have tools that can continuously do builds, continuously do integration, and continuously do deployments. The tool I'm going to show you is called Jenkins. Now Jenkins is a DevOps E like continuous integration tool that performs or builds for us. So over the next couple minutes or so is I'm going to give you a brief overview on Jenkins. First of all, Jenkins is free. It's an open source project. And you can download it, install it, and just about anything. So whatever you are running, you're most likely to get Jenkins to run on it.   
  
So what we're going to do here is I'm going to show you a demo on creating a new Jenkins job. As you can see, I have some already set up here. But let me go to the left, click **New Item**, and we will build a new Jenkins job. And I'm going to give you the cliff notes because I just want you to see just kind of the main features of Jenkins. So here what I'm going to do is I'm going to build a **Freestyle project**, which I will call DevOps Jenkins Demo. And notice, we could do Maven, external, multi-configuration, or build a job based on an existing one we already have. But we're going to stick with **Freestyle project**. Click **OK**. And I'm just going to show you some options here. We could put in our Description. And let me show you the options for creating this item. Then navigate down. This is a big one here. Source Code Management – by default, we have **CVS**, **CVS Projectset**, and **Subversion**, or **None** meaning that in this case we have our Java code local. It's important to point out though that we do have some plugins that we could use to actually use public repositories, such as if you use Git. We could use Git as our source code management tool or Bitbucket. Bitbucket uses Git and some other ones in the background for a source code management.   
The presenter clicks the New Item link and the area to the right now shows the Item name text field. Below the text field are the Freestyle project, Maven project, External Job, Multi-configuration project, and Copy existing Item radio buttons. Below the "Copy existing Item" radio button is the "copy from" text field and the OK button. The presenter selects the "Freestyle project" radio button and types "DevOps Jenkins Demo" in the Item name text field. He then clicks the OK button. The area to the right now has the Project name text field with the default entry of DevOps Jenkins Demo. Below the Project name text field is the Description text field. The presenter adds the text "DevOps Jenkins Demo" in the Description text field. The area to the right also includes the Source Code Management, Build Triggers, Build, and Post-build Actions sections, among others. The "Source Code Management" section consists of the None, CVS, CVS Projectset, and Subversion radio buttons. The None radio button is selected by default.   
  
This is important because what we can do is we could automate builds based on the code that's in those repositories – either public or private, which is very important if you work in a distributor team, especially a DevOps team because you could have developers all over the world contributing to that build and checking their code in the Git. And you could actually build a project based on what their code is. So let me navigate down a little bit further and explain to you build triggers. We could build after other projects are built periodically, or we could pull the source control management tool. Now this is important, especially these two right here. We can schedule a built. We could say everyday we want you to go to this repository and build this application. We could say every minute go to that repository, build that application, or this one here – which is the most interesting – which is pulling the source control management tool. How about this? How about we want to schedule an automatically fire a build off every time the source is changed within our source control management tool? Pretty cool because that way what we can do is we can actually automate or continuously integrate any changes that we have based on code that's checked in. Now obviously, that's going to lead to some QA or some UAT issues. But I'll leave it to you to figure out exactly how you want to handle those.   
The Build Triggers section has the "Build after other projects are built," "Build periodically," and "Poll SCM" checkboxes.   
  
Now moving on, we also have some additional build steps where we can actually here **Invoke Ant** scripts, shell scripts, Windows batch commands. And we have post-build actions where we can **Build other projects** for testing. We can publish JUnit test results or this one here – provide an **E-mail Notification**. So these are all the different parameters you can set if you want to automate your build process. And to do so, we are using Jenkins to actually continuously integrate any code changes that we might have that we may build them and maybe deploy them into production.   
The Build section has the "Add build step" drop-down list box and the Post-build Actions section has the "Add post-build action" drop-down list box. The presenter clicks the "Add build step" drop-down list box to display four drop-down list box options: Execute Windows batch command, Execute shell, Invoke Ant, and Invoke top-level Maven targets. He then clicks the "Add post-build action" drop-down list box to display seven drop-down list box options: Aggregate downstream test results, Archive the artifacts, Build other projects, Publish JUnit test result report, Publish Javadoc, Record fingerprints of files to track usage, and E-mail Notification.

[Back to top](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#top)

# Using DevOps for Maintenance and Release Scheduling

## Learning Objective

After completing this topic, you should be able to

* *define the steps for DevOps software releases and maintenance scheduling and recognize DevOps release tools such as Jenkins*

## **1. DevOps software releases and its tools**

Software maintenance in a traditional environment is cumbersome and expensive. Some organizations spend more money on software maintenance than on software development. Sustainment engineers are usually responsible for scheduling software maintenance. Maintenance is usually handled within maintenance windows. These windows are very narrow spans of time, where an application may be taken offline. Maintenance often leads to downtime depending on what is being maintained. The DevOps continuous improvement paradigm replaces traditional maintenance. If applications are continuously integrated, there is no longer a need for maintenance stage. Maintenance is now more collaborative and less expensive. Release scheduling or builds is continuous in DevOps. Software releases can be in the order of hundreds a day. Most traditional organizations may deploy an application every few months or even years. The volume and velocity of software releases has increased exponentially. Builds and deploys are no longer clumsy manual processes. This increased release scheduling has led to the wholesale automation of software releases. Automation avoids the problems that occur in manual releases by building inconsistency into the build-deploy process. Continuous integration allows DevOps personnel to improve the process of the build release cycle rather than concentrate on each individual application being deployed.   
  
There are many DevOps release automation tools. Of course, some are free. There are also some vendor-specific tools customized for a language or a platform. They speed up the automation release cycle. Applications can be built and deployed from a mouse click. Errors are reduced and the release process is standardized and repeatable. Automation reduces the cost of releasing software. Automation tools can be used to manage complex release and versioning tasks. Jenkins is a continuous integration tool. Written in Java, the project is an offshoot of another popular tool, Hudson. Jenkins is free. Plugins allow Jenkins to be run in non-Java environments. Like comparable tools, Jenkins automates the build process. The concept of continuous integration has replaced manual builds. Jenkins monitors execution of the automated build steps. Builds can be initiated in a number of ways. Builds can be scheduled automatically through a commit to source control or even after another project has been built. Jenkins can run as a command line application or within a web container.   
  
There are many different continuous integration tools. They pretty much run the gamut and what they can do and what they support, but they're loosely organized into three categories: vendor, platform, and language. Buildbot is used for Python-based software development. Buildbot started as a light weight alternative to Tinderbox. Mozilla is a flagship user of Buildbot. CruiseControl is used for Java and .NET applications. It includes many plugins and support a variety of source and build options. Team Foundation Server is a Microsoft continuous integration engine and is at the core of Microsoft's Application Lifecycle Management solution, ALM. CABIE is an open source Perl-based automated build and integration environment. CABIE builds jobs based on build info stored in a MySQL database and will support just about any command line build.

[Back to top](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#top)

# Using HipChat for Distributed Team Management

## Learning Objective

After completing this topic, you should be able to

* *describe how HipChat is used to manage geographically separated teams*

## **1. Managing teams with HipChat**

HipChat is a market leader in DevOps communication tools. Released in 2012 and owned by Atlassian, HipChat has quickly gathered market acceptance. Traditional communication tools, such as e-mail, are cumbersome. DevOps communication tools attempt to replace voicemail, e-mail, and even text messaging. DevOps requires nimble team communication skills, such as HipChat. Currently HipChat is based on the freemium model meaning most features are free, and extended features are available at 2 USD a month. Like most DevOps tools, HipChat has a small footprint and is run anywhere. HipChat is extremely easy to use and configure. HipChat is built for distributed team management. Throughout all of its coolness, it's not a toy. HipChat attempts to coordinate multiple modes of communication into one application. HipChat features include screen sharing and video calling. Security features are integrated into HipChat. Secure conversations occur in 256-bit SSL. HipChat organizes communication into groups and rooms. You start in the HipChat lobby where you can enter rooms to join a conversation or create a room of your own. Conversations within rooms can be identified by its title and its topic. Private rooms and conversations are not visible in the lobby. You can only see open rooms. HipChat has a few features that make it unique, such as HipChat ensures delivery to team members who are offline.   
  
Basic HipChat is free and includes instant messaging, group chat, and file sharing. HipChat keeps track of everything you upload and any links that you share. You also get to see the history of everything that was posted in a room. Sharing files is simple, and HipChat supports drag and drop. When using common media files, HipChat will even allow you to preview your upload. Like most DevOps tools, HipChat is very scalable. HipChat plus adds screen sharing at $2 a month. Dedicated server – cloud and enterprise – are available. HipChat server prices scale from $10 a year for 10 users to $72,000 a year for 5,000 users. HipChat is designed for IT professionals. It is built for teams. It can be privately hosted. Many features were incorporated with the software developer in mind, such as a source code editor. This means a code can be shared and highlighted. Communication between software types often does not occur in real-time or has to be triggered. Messages can be automated, such as system messages, to support personnel. There are also management hooks. Communication and collaboration can be monitored by management. Also HipChat offers real-time, all-the-time team communication.   
  
HipChat integrates seamlessly into other DevOps tools – Atlassian owns BitBucket, Confluence, and JIRA. So expect better than average integration with those tools. The following are a few examples of other DevOps tools that integrate well with HipChat. Integration with GitHub notifies team members of new tasks, code commits, et cetera. Integration with UserVoice notifies team members of new support tickets and reviews. Integration with TeamCity notifies the team of new or failed tests. Integration with HubBot allows team collaboration when performing deployments.

[Back to top](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#top)

# Using GitHub for Collaboration

## Learning Objective

After completing this topic, you should be able to

* *specify how collaboration occurs with GitHub*

## **1. How does GitHub facilitate collaboration**

GitHub is the world's largest code repository. Over the years, public source code control has gained widespread acceptance. GitHub allows teams to create and share software and is based on Git. Git allows Distributed Version Control and Source Management Control. It's been around for years and is the actual source repository for other source control tools such as BitBucket. GitHub is the web-based GUI that runs over Git. GitHub is built for the public. It's also the world's largest code repository claiming over 24 million public repositories. Collaborative features include wikis and integrated bug tracking. GitHub is part code repository and part social network. All the cool kids are there. Most open source project can be found on GitHub including Linux and Amazon Web Services. The whole GitHub application is built around getting public exposure to your code and asking for collaboration. You have the ability to follow other GitHub users. Users can also follow entire projects. Developers friend each other like on Facebook. Developers can also send requests to contribute to other projects. The main functionality of GitHub is forking or copying an entire code repository from one account into another. This effectively allows you to take an authorship of an entire project. Since Git encourages documenting small code changes, other developers can look to see how previous programmers solve tricky problems.   
  
Recently GitHub has grown beyond source code management. Collaboration is not just for developers or code. Any document or group of documents can be versioned. GitHub is evolving and drawing non developer users. Urban planners use GitHub to share documents such as historical maps and engineering surveys. Municipalities are storing laws on GitHub. Architects and engineers use GitHub for document and design collaboration. GitHub has become the library of congress for code and document repositories. Like most DevOps tools, GitHub has a free version. The free version of GitHub allows for an unlimited number of public repositories. Public repositories can be viewed by anyone. Private repositories can only be seen by you and your collaborators. The number of private repositories available is determined by your pay plan. Plans include 1 gigabit of storage. An additional 50 gigabit is available for $5 a month. GitHub runs everywhere. Most traditional and mobile platforms are supported. All GitHub users automatically get a personal account. As projects get larger, individual accounts can transfer into an organization account as your project adds collaborators.   
  
Future versions will support Large File Storage – LFS versioning. Git Large File Storage will replace large files such as videos, audio files, and high-resolution pictures with the pointer inside Git. The actual contents of the file will be stored on a remote server such as GitHub Enterprise or github.com. Other planned features are "a what you see is what you get" – web-based text editor – Easel – is being incorporated into GitHub. GitHub will be offering more blog designs, currently it only has 11. Also, GitHub is integrating drag and drop. GitHub will continue to migrate toward social network model.

[Back to top](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#top)

# Sharing Software Issues with JIRA

## Learning Objective

After completing this topic, you should be able to

* *describe how JIRA is used to log and share software issues*

## **1. Using JIRA for sharing software issues**

In this demonstration, I'm going to show you a product called JIRA. Now what JIRA is? This is a DevOps communication tool that allows teams to log software issues. You can log issues that have to do with development, perhaps log issues that have to do with deployment, a bug fix – really anything could be entered in JIRA. And what makes JIRA very interesting is it integrates well with other DevOps tools such as HipChat and Bitbucket since they're made by the same company. It also integrates really well with probably two or three dozen other really nice DevOps tools. Let me show you JIRA. JIRA starts at the dashboard. At the dashboard, you get to see in your right-hand side all the issues that are assigned to you. And each of these icons here has the status and the priority of each issue. If you look to the right in the bottom, you'll see your Activity Stream. Your Activity Stream shows all of the issues that have been logged in a streaming fashion by your organization. Now moving up from the Dashboards, go to Projects. And in Projects, you can look at what your CURRENT PROJECT is or all of the projects that you've been set up to view. Here I'm going to navigate to a CURRENT PROJECT called **eCIT (ECIT)**.   
The JIRA web site is shown. The web site has six drop-down menus: Dashboards, Projects, Issues, Service Desk, Structure, and Epics. Next to the drop-down menus is the Create button. Below the menus, the System Dashboard page is shown. The System Dashboard page has three sections: Introduction, Assigned to me, and Activity Stream. The presenter discusses the Assigned to me and Activity Stream sections. The presenter clicks the Projects drop-down menu to display two drop-down menu options, Current Project eCIT (ECIT) and View All Projects. The presenter clicks the Current Project eCIT (ECIT) drop-down menu option to navigate to the eCIT page.   
  
Now on eCIT we have the Summary. We have some useful links here that we can add. And we also have an Activity Stream for this specific project. Now these...here are different issues that I've either added or I've been working on. Now to drill down into these issues, you can navigate down here where you can look at **My Open Issues**. And in the left-hand side, you'll see all the issues that have been assigned to you. And on the right, you get the detail for those issues. Now, when you work on these issues, you can change the state from in development, ready to test, deployed, retest. And you can set the priorities such as preempt our work, high priority, medium priority, or low priority. And there's nothing that's really low priority anymore.   
The presenter clicks the Issues drop-down menu to display several menu options. He then selects the "My Open Issues" menu option from the Issues drop-down menu to navigate to the "My Open Issues" page. The presenter discusses the "My Open Issues" page.   
  
Now what is important about this tool is, like I said, it integrates really well with other DevOps tools. For example, if I was using Bitbucket for source control, I can have Bitbucket and JIRA communicate with each other. So as I'm checking code out of Bitbucket and maybe committing code in a Bitbucket and putting comments of the code in Bitbucket, I can actually have JIRA actually to log those issues so I can have those different tools communicate with each other. And also the same is with HipChat. If I use HipChat for team collaboration, I could have HipChat talk to JIRA and JIRA actually talk to Bitbucket. And, if I want to get really crazy with this, I could even get Jenkins to actually get the builds and get the actually code commits from Bitbucket. And actually do the build based on what I'm entering in JIRA and what the code developers are actually putting into back in a Bitbucket. So couple more things here. If you want to look at Issues, like for example that I've opened, I can navigate to **My Open Issues**. If I wanted to look at, for example, all the stuff that was reported by me, I can navigate down to **Reported by Me** and it will show all of the Issues that I actually entered. And then the right, it will show the Status and the Details of that specific issue. JIRA is a software issue-tracking tool that works really well with other DevOps applications and works really well with communicating different issues and the status of each issue.   
The presenter clicks the Issues drop-down menu, and selects the "Reported by Me" menu option to open the "Reported by Me" page. He then discusses the contents of the "Reported by Me" page.

[Back to top](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#top)

# Aligning Teams Using Confluence

## Learning Objective

After completing this topic, you should be able to

* *use Confluence for parallel team management*

## **1. Parallel team management with Confluence**

Confluence is a team collaboration software suite. Originally released in 2004, it is one of the more established DevOps tools. Confluence is a great example of a DevOps tool that has been around before the actual coining of the expression. Confluence is tightly bound with other DevOps tools such as Bamboo and JIRA. This allows your Wiki to do more as it's integrated with other Cloud applications that you use. At its core, Confluence is a team Wiki. Confluence creates a place for your team to share and organize work. All of the features of Confluence allow your team to turn your pages into rich dashboards where all your information is available at a glance. Confluence allows for planning and organizing meetings. It has professional layout and editing features. Documents can be created, viewed, and shared in one place. Hierarchies of pages and cross-page references can easily be created. Useful and appealing templates – blueprints – can be used for workflow documents. Teams can share ideas and provide feedback. Spaces, blogs, and pages can be created and edited. Confluence allows team tasks to be added to pages and blogs. Confluence is very visually pleasing. Because of its effortless look and feel, Confluence has generated interest throughout the entire company, not just from IT folks.   
  
Confluence is free to small teams of five people or less. Pricing is tiered – 10 users at $10 a month to 2,000 users at a $1,000 a month. Confluence can be hosted on your own physical servers. Private – you host server versions of Confluence – can be purchased for a little as 10 USD for 10 users up to 24,000 USD for 10,000 users. Cloud-based Confluence is available through à la carte product offerings. Atlassian Cloud also offers integration into Bamboo and JIRA. Confluence server runs on Linux and Windows and has recent support for Mac. Server versions have robust drivers to connect to legacy databases. Confluence is optimized for mobile devices and is primarily run through mobile browsers. Confluence is supported by Apple – Mobile Safari – and Android. Since Confluence is a pure Java application, mobile devices must have the correct JDK or JRE installed and accessible by Confluence. Apple and Android apps are being developed. Desktop support includes Internet Explorer, Firefox, and Safari. Mobile Confluence is priced the same as the server version. Robust backend database integration provides mobile access to legacy data. Confluence supports PostgreSQL, MySQL, Oracle, and SQL Server.   
  
The National Hockey League uses Confluence for issue tracking and documentation. The NHL also uses JIRA for bug tracking and Crowd for user feedback. Business people at the NHL like to use Confluence to create requirements and the definitions of new products. Developers like Confluence for designing software and delivering documentation to end users. The Dow Jones & Company uses Confluence as a team integration platform. OpenDNS uses Confluence as a substitute for e-mail. ShopLocal uses Confluence to share technical documentation. Sega uses Confluence to create and share game development documentation. Almost everyone at Sega uses Confluence and uses it every day as a primary source for company information. Game leads blog to their Confluence page every day.

[Back to top](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#top)

# Sharing Code with Bitbucket

## Learning Objective

After completing this topic, you should be able to

* *use Bitbucket for code sharing and versioning*

## **1. Code sharing and versioning**

Bitbucket is a web-based source control...version control tool. Bitbucket supports both Git and Mercurial revision control systems. This is an important point as Bitbucket is the largest source control versioning tool that supports both the popular Git and the more refine Mercurial communities. Considered a direct competitor of GitHub, it is the second most popular free source control tool ahead of Stash and script. Owned by Atlassian – who also owns both JIRA and HipChat – Bitbucket started out as an independent project in 2008 and supported Mercurial only. Git support was added in 2011 just after being purchased by Atlassian. Bitbucket is open source and written in Python. Bitbucket and GitHub have significant differences. GitHub has more social networking features, not exactly a detriment but it's an important distinction. All of the glam and the reality show based "look at me" vibe of GitHub might turn off more quiet, refine developers who turn to Bitbucket. Bitbucket allows five private repositories in their free version. GitHub allows private repositories with a paid subscription. Bitbucket has tighter integration with other DevOps tools, such as Atlassian-owned JIRA and Confluence. Both are open source and have relatively similar pricing plans. Both have decent free plans with plans becoming more expensive as the use of private repositories grows.   
  
Bitbucket is more focused toward enterprise developers and more private collaborative development. Teams can be built quickly. Bitbucket is free for up to five users on your team and only 1 USD for each additional user. Nonprofit and university accounts are also free and receive unlimited private and public repositories. GitHub favors public collaborative development and attracts coders looking for friends and to attach their name to an open source project. Because of this, Bitbucket does not have notable projects, such as Linux. Bitbucket has one million users, GitHub has four million. Bitbucket also has more authentication support, such as Twitter and Facebook. Bitbucket allows code reviews on commits. Branches and pull request can be created in the same repository. Bitbucket also allows you to create and manage multiple file code snippets, text, and multimedia assets. Most support is for the traditional desktop however. Additional Apple and Android apps are being developed. GitHub repositories can be migrated to Bitbucket. Bitbucket, as well as GitHub, can be hosted locally. Bitbucket's behind-the-firewall Git repository solution is called Stash. Stash allows you to create and manage repositories, set up custom permissions, and connect via LDAP. New features of Bitbucket include upgraded Diff functionality and to add a bit of fun, support of emojis.   
  
Many developers choose to use GitHub and Bitbucket. If you use Git as your ultimate code repository, projects and teams can move between the two tools pretty easily. Code that needs public exposure is sourced on GitHub. Public exposer adds to the marketing of the project. Closed enterprise code is sourced on Bitbucket. Private repositories do not get the attention of public repositories. For that the developers in Bitbucket sigh a collective "So what?" Projects that need tight integration into other DevOps tools tend to use Bitbucket. Projects that need more web GUI support use GitHub albeit both of the last points may be subjective depending on the application.

[Back to top](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#top)

# Managing Cross-Platform Development with DevOps

## Learning Objective

After completing this topic, you should be able to

* *describe how DevOps is used to manage cross-platform development issues*

## **1. Cross-platform development issue**

DevOps offers a plethora of tools used to automate task management. Task management tools are used for everything – from checking e-mail to viewing customer feedback on a deployed website. Task management tools allow the execution of commands in a target machine. The back end of these tools usually contains an API that automates operations that previously were manual. Task management tools automate simple redundant tasks. Tools like Asana automate project management by processing tasks and generating timelines. Tools usually run on a single device and are generally not good at multiple machine coordination. They have poor performance in working in hybrid environments. Any customization is expensive and may be impossible. Generally, they don't have out-of-the-box business logic. Cloud and container tools are used to manage virtual machines. Docker is a container tool that wraps up you application and allows it to be moved and deployed anywhere. Tools offer on-demand environments and delivery pipelines. An application can be staged and continually be piped and installed in a container to be deployed in the cloud. This system literally encapsulates applications with all of its dependencies and isolates it from the rest of the world. This also provides container standardization environments. Some of these tools don't adapt well to existing applications. Also, by encapsulating existing code and dependencies, container tools often automate the problem rather than fix it.   
  
System provisioning and configuration tools define the state of a system. In a nutshell, this means that configuration tools ensure that the thousands of network configuration options are set properly. They also ensure that all machines are in a predictable state – like they're up and running. Tools provide initial system configuration and services. They also monitor the intended versus actual state of a machine and can make configuration changes on the fly. Tools are not designed to handle application deployments, but to ensure that the network that the application is deployed upon is configured correctly. Continuous application integration or CI tools create application builds from source artifacts. Artifacts include build scripts, such as legacy tools like ANT. Tools provide code testing and analysis functionality. They often chain and distribute testing and build tasks. The concept of continuous integration and continuous delivery assumes that the software that made code branch design is also in a deployable state. This makes application deployment a very rapid process. Tools maintain production candidates from a main code branch. Continuous integration tools are generally not used for coordination across multiple machines.   
  
Pipeline orchestration tools allow the definition and the sequence of the delivery process. Applications are developed and advanced through incremental stages of software development and deployment readiness. Pipeline tools provide visibility into the application delivery process. This delivery process allows development and operations staff to continually evaluate the way software is promoted. They also provide a standard process in which applications are deployed. Pipeline orchestration tools can be used to define the process used by other tools that perform the actual deploy. Indeed many pipeline tools feed into continuous delivery systems. Tools also provide a roadmap for each production application. Pipeline orchestration tools also provide a roadmap for each production application.

[Back to top](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#top)

# Exercise: Set up DevOps Processes and Tools

## Learning Objective

After completing this topic, you should be able to

* *describe the software development life cycle within an organization and be able to recommend DevOps processes and tools*

## **1. SDLC and DevOps processes and tools**

In this exercise, you will describe the term DevOps, identify the stages of the software development lifecycle, identify different functional groups of DevOps tools, identify the two groups referenced by DevOps, describe how DevOps changes the role of the two groups mentioned above. Now pause the video and perform the exercise and come up with your best answer. When you're finished, resume the video to see my answers.   
  
Welcome back. Do you have your answers? Compare your answers to my solution. Now, remember, these questions are subjective – meaning is more than one correct answer. DevOps is an agile software development methodology intended to speed up the software development lifecycle. It attempts to break down silos and encourage collaboration between groups. DevOps has its own ecosystem, which helps fuel its growth. Now, if this was your answer, congratulations. If your answer was close, you're probably pretty good too. Remember there is not one single definition of DevOps. The stages of the software development cycle are requirement gathering, design, development, testing and build and deployment. Now again, if your answer was close, give yourself full credit for this answer. There are some additional stages depending on what kind of shop that you are running.   
  
As discussed in this video, the groups of DevOps tools are team management, collaboration, issue tracking, team alignment, code sharing, and cross-platform development management. As far as DevOps tools go, there are no discrete categories of them. But these are the ones that we discussed in the video. So give yourself credit if you got most of them. The point is as you recognize the DevOps tool when you see it and you could pretty much categorize which group it belongs to, the two groups referenced by DevOps are development and operations. Also, as discussed, it's important to point out that DevOps actually touches more groups than development and operations such as QA and security. DevOps changes the role of the two groups by encouraging collaboration. Development or developer is performing traditional operations tasks, automating workflows through tools, and sharing common goals such as continuous integration.

[Back to top](https://library.skillport.com/courseware/Content/cca/os_devo_a01_it_enus/output/html/course_transcript.html#top)

