## What is CND?

[**Computer Network Defense (CND)**](https://www.bitlyft.com/glossary-of-terms/#accord-a-2) is a form of cybersecurity for the securing of military and government computer systems. Like everyone else in the world, national agencies also have to secure their systems against malicious cyber attacks.

We live in a highly technological era, with computers and other technology being used for good all over the world. People use computers to create new things for people everywhere to enjoy. People use computers to write stories, make videos, and even build exciting new things online, like websites and video games.

Unfortunately, like every other good thing in the world, computers can also be used to do unscrupulous things, like hack into corporate networks and steal customer data and intellectual property. Hackers have been doing this since the internet was invented, forcing companies and webmasters to adapt their security practices with the changing times.

All over the world, militaries and government agencies are using computer network defense to secure their systems and protect their national security by keeping hackers out of highly important systems. This helps to make critical infrastructure and other important national systems safe from those who wish to do them harm.

What are the Challenges With CND?

**Let’s face it:** hackers and those who seek to use technology for their own nefarious gains are here to stay. With this ongoing problem, numerous new challenges arise every year to cybersecurity professionals who want to make sure systems are secure for the organizations and people who depend on their continued operations, as well as for military operations and government agencies.

These challenges present new obstacles for security professionals to adapt to and overcome. A key aspect of computer network defense is the ability to adapt to new problems and threats *as they arise*. This is why a good [incident-response plan](https://www.bitlyft.com/performing-an-it-risk-assessment/) with clearly defined cybersecurity protocols is of vital importance to professionals in security everywhere.

These are just a few of the issues that continue to plague security professionals to this day:

* **Malware**. [Malware](https://www.bitlyft.com/how-saas-security-can-fix-your-blind-spots/)basically just means “malicious software,” and it is everywhere. Hackers can employ various types of malware to achieve some of their nefarious goals across all different sectors. Whether they want to break into a corporate network, steal customer data, take over a website, or gain access to privileged accounts. There are all different types of malware. [Ransomware](https://www.bitlyft.com/the-true-cost-of-samsam-ransomware-in-the-hidden-threat/) is a type of malware that can encrypt notebook and desktop computers and demand a payment for their decryption. Remote access trojans can lie dormant in systems and then completely take them over, giving the hacker the sort of control he would have as if he were physically sitting at the keyboard.
* **Humans**. Like it or not, human users are still the largest vulnerability present to any protected system. A computer network can be very secure, but the humans who protect it and use it are not infallible, and are susceptible to human failings. If a hacker can influence someone to click on a sketchy link or install a malicious program, they can still gain access to the system. [The human vulnerability](https://www.bitlyft.com/can-automated-threat-detection-improve-your-security-response/) is one huge reason why every organization (especially governmental agencies) should make cybersecurity a key point to teach every employee. If employees are well learned about their organization’s cybersecurity practices, they are less likely to fall victim to hacking schemes.
* **Phishing**. [Phishing](https://www.bitlyft.com/dont-get-caught-up-in-the-latest-phishing-attack/)(and spear phishing) is still one of the main ways hackers find their way into protected systems. All it takes is an email appearing to be from an important person within the organization, and an unsuspecting employee could click on a link or install a malicious file from an email. This is all it takes for hackers to gain access to systems and do monumental damage.
* **Finances**. The sad truth is that a lot of organizations just don’t put enough money into their cybersecurity systems. It is important for organizations to include a healthy budget for a proper security program, educating all users (not just the security professionals) about the importance of good cybersecurity practices. Unfortunately, some organizations don’t obtain funding for a proper cybersecurity program until after they’ve fallen victim to a data breach or hack.

These are just some of the challenges facing CND today. These remain some of the biggest and consistent issues in an ever changing sea of digital threats on the rise today. Knowing some of these challenges facing your CND strategy can help you to adapt it and make it better as new challenges present themselves. This is why it is so important for security researchers to stay alert to new threats popping up in the news year after year.

Best Practices With CND

There are some proven methods for making sure CND approaches are rock solid for organizations everywhere. Adhering to these best practices is the best way to ensure your network stays protected from hackers and those with malicious intent.

* **Utilize a firewall**. One of the first and best defenses your network can employ is a firewall. A firewall acts as a barrier between an organization’s valuable data and the criminals trying to steal it. Firewalls provide an extra layer of security to the layers already in place. At the same time, your assets don’t always stay behind the corporate firewalls, so it is also important to have a good understanding of what those assets are doing while not behind a firewall.
* **Visibility.** Having visibility into and across your entire network is critical today. You need to know what traffic is on your network, should it be there, and has it always been there? You also need to have [solid visibility into your cloud products](https://www.bitlyft.com/cloud-security-as-a-service/) you use today, Office 365, Google Gmail and Salesforce.com to name a few. Knowledge and awareness of who is accessing your systems, when they are and if they are the intended user accessing those systems. If they are not the intended user, you need to be alarmed immediately.  Having this type of visibility is a minimum requirement with today’s technology landscape.
* **Document and outline your cybersecurity practices**. Have every plan, from[incident response](https://www.bitlyft.com/what-is-security-incident-response-plan-2/) to the types of plans in place for different types of attacks, well documented and outlined. This will help in the review of CND practices, as well as help adapt these security policies to new problems that arise over time. Having a solid and adaptable plan for security is one of the best ways that organizations can keep their systems air-tight.
* **Have a plan for mobile devices**. If your organization relies on mobile devices such as smartphones and tablets, it is important to have a plan to support those as well. Keep an outline of who is issued which device, and ensure that every device is up to date with the latest software and security patches. Mobile devices can be updated through their app stores or settings dependent on which operating system they run. Check for updates for Google’s Android using Google Play Protect in the Google Play Store, or through the Android system settings. Apps for Apple’s iPhones can be updated through the Apple App Store, and system updates can be applied through the iPhone system settings by clicking “Check for Updates.”
* **Educate employees on the importance of cybersecurity best practices**. Make sure all employees are aware of the organization’s security practices. This will help make sure members of staff aren’t influenced by phishing scams to click on unknown links or install malicious files onto their systems.

## Table of Contents

[What Is a Pipeline in DevOps?](https://www.simplilearn.com/what-is-pipeline-in-devops-tools-implementation-article#what_is_a_pipeline_in_devops)

[Components of a DevOps Pipeline](https://www.simplilearn.com/what-is-pipeline-in-devops-tools-implementation-article#components_of_a_devops_pipeline)

[What is Pipeline in DevOps: The Phases of a DevOps Pipeline](https://www.simplilearn.com/what-is-pipeline-in-devops-tools-implementation-article#what_is_pipeline_in_devops_the_phases_of_a_devops_pipeline)

[DevOps](https://www.simplilearn.com/devops-revolution-article" \o "DevOps" \t "_blank) is a popular application design philosophy that merges development and operations, hence the clever name. However, there are many terms relating to DevOps, and it’s helpful to sometimes look at one of them and explore it in-depth. By breaking down a concept like DevOps and focusing on one element at a time, we can gain a greater overall understanding of it, which in turn helps us get more out of the process.

So, to get to know DevOps better, we're looking at pipelines in DevOps. This article will answer the question, "What is a pipeline in DevOps?". Additionally, we will explore the components, phases, and stages of the DevOps pipeline and even dedicate some time to explaining what the [Azure DevOps](https://www.simplilearn.com/azure-devops-article) pipeline is all about.

## What Is a Pipeline in DevOps?

A DevOps pipeline is the set of automated processes and tools that the development and operations teams use to compile, construct, test, and deploy software code faster and easier. However, the term "pipeline" isn't an exact fit; it's more like an assembly line. For instance, an automobile that goes through the factory assembly line undergoes continuous assembly. Workers first build a chassis, add the engine, doors, tires, seats, and instrument panel, and finish it with exterior paint.

The DevOps pipeline works like that, starting with writing the code and then running tests to find bugs, errors, typos, and redundancies. [DevOps teams](https://www.simplilearn.com/core-components-devops-team-needs-to-be-successful-article) then put fixes and patches to address the issues, test them some more, and finally release the working product to users.

## Components of a DevOps Pipeline

The DevOps pipeline is composed of seven components:

* [Continuous Integration and Continuous Delivery](https://www.simplilearn.com/tutorials/jenkins-tutorial/ci-cd-pipeline): These two components are typically mentioned together, usually referred to as CI/CD or a CI/CD pipeline. Continuous integration means the system frequently integrates new code changes into the central repository, usually a few times per day. This process makes it easier to merge different code changes and spot bugs.

The continuous delivery aspect means incremental deliveries of software and updates to production. CD helps developers automate the whole software release operation and increase how frequently they release new features.

* Continuous Testing: DevOps personnel use continuous testing to perform automated tests on any code integrations accumulated during the continuous integration (CI) phase. CI ensures high-quality app development and evaluates the release's risks before sending it to delivery.
* [Continuous Deployment](https://www.simplilearn.com/tutorials/devops-tutorial/continuous-delivery-and-continuous-deployment): This component is often blurred with continuous delivery, although both are very different parts of the process. Continuous deployment follows continuous delivery. Any updates that successfully make it through the automated testing phase get automatically released into production. This way, continuous deployment allows multiple production deployments on a given day.
* Continuous Monitoring: Continuous monitoring validates the environment’s stability and verifies that the applications are doing what they’re designed to do. In addition, the operations teams monitor the applications and systems, keeping an eye on the latter's performance.
* Continuous Feedback: Continuous feedback is often overlooked, which is regrettable because DevOps teams need constant feedback to ensure that the app does what everyone (the developers, stakeholders, and customers) expect it to. Everyone needs to be on the same page, and that’s what continuous feedback does.
* Continuous Operations. This component does exactly what the name implies: maintaining a 24-hour, seven-day-a-week operation with little to no planned downtime. The ultimate goal of continuous operations is to ensure that end-users won't suffer interruptions due to any hardware or software changes. It's an expensive initial investment but pays for itself in the long run because it prevents costly production losses.

## What is Pipeline in DevOps: The Phases of a DevOps Pipeline

Now that we’re familiar with the components of the DevOps pipeline, it’s time to look at the phases, or stages, of the pipeline.

* Develop: The developers write the software code and then push it into the source control repository, after which the source code integration occurs.
* Build: In the next stage, the application is built with the integrated source code from the previous phase’s source code repository.
* Test: In this phase, testers execute various tests (functional, system, unit) on the build created in the previous stage. If the tests reveal issues, they’re kicked back to the developer to be resolved.
* Deploy: This final stage sees the deployment of the final version, conducted when the production environment is created and configured.

Git - Basic Concepts

Version Control System

**Version Control System (VCS)** is a software that helps software developers to work together and maintain a complete history of their work.

Listed below are the functions of a VCS −

* Allows developers to work simultaneously.
* Does not allow overwriting each other’s changes.
* Maintains a history of every version.

Following are the types of VCS −

* Centralized version control system (CVCS).
* Distributed/Decentralized version control system (DVCS).

In this chapter, we will concentrate only on distributed version control system and especially on Git. Git falls under distributed version control system.

Distributed Version Control System

Centralized version control system (CVCS) uses a central server to store all files and enables team collaboration. But the major drawback of CVCS is its single point of failure, i.e., failure of the central server. Unfortunately, if the central server goes down for an hour, then during that hour, no one can collaborate at all. And even in a worst case, if the disk of the central server gets corrupted and proper backup has not been taken, then you will lose the entire history of the project. Here, distributed version control system (DVCS) comes into picture.

DVCS clients not only check out the latest snapshot of the directory but they also fully mirror the repository. If the server goes down, then the repository from any client can be copied back to the server to restore it. Every checkout is a full backup of the repository. Git does not rely on the central server and that is why you can perform many operations when you are offline. You can commit changes, create branches, view logs, and perform other operations when you are offline. You require network connection only to publish your changes and take the latest changes.

Advantages of Git

Free and open source

Git is released under GPL’s open source license. It is available freely over the internet. You can use Git to manage property projects without paying a single penny. As it is an open source, you can download its source code and also perform changes according to your requirements.

Fast and small

As most of the operations are performed locally, it gives a huge benefit in terms of speed. Git does not rely on the central server; that is why, there is no need to interact with the remote server for every operation. The core part of Git is written in C, which avoids runtime overheads associated with other high-level languages. Though Git mirrors entire repository, the size of the data on the client side is small. This illustrates the efficiency of Git at compressing and storing data on the client side.

Implicit backup

The chances of losing data are very rare when there are multiple copies of it. Data present on any client side mirrors the repository, hence it can be used in the event of a crash or disk corruption.

Security

Git uses a common cryptographic hash function called secure hash function (SHA1), to name and identify objects within its database. Every file and commit is check-summed and retrieved by its checksum at the time of checkout. It implies that, it is impossible to change file, date, and commit message and any other data from the Git database without knowing Git.

No need of powerful hardware

In case of CVCS, the central server needs to be powerful enough to serve requests of the entire team. For smaller teams, it is not an issue, but as the team size grows, the hardware limitations of the server can be a performance bottleneck. In case of DVCS, developers don’t interact with the server unless they need to push or pull changes. All the heavy lifting happens on the client side, so the server hardware can be very simple indeed.

Easier branching

CVCS uses cheap copy mechanism, If we create a new branch, it will copy all the codes to the new branch, so it is time-consuming and not efficient. Also, deletion and merging of branches in CVCS is complicated and time-consuming. But branch management with Git is very simple. It takes only a few seconds to create, delete, and merge branches.

Git - Life Cycle

In this chapter, we will discuss the life cycle of Git. In later chapters, we will cover the Git commands for each operation.

General workflow is as follows −

* You clone the Git repository as a working copy.
* You modify the working copy by adding/editing files.
* If necessary, you also update the working copy by taking other developer's changes.
* You review the changes before commit.
* You commit changes. If everything is fine, then you push the changes to the repository.
* After committing, if you realize something is wrong, then you correct the last commit and push the changes to the repository.

Shown below is the pictorial representation of the work-flow.

