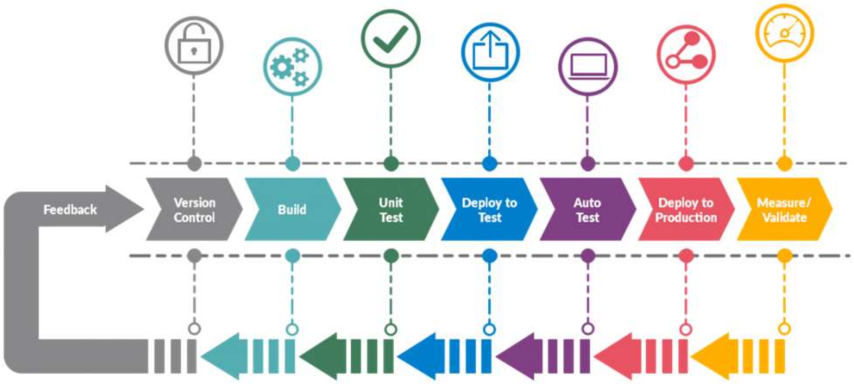
DevOps Pipeline



Feedback

Version Control

Build

Unit Test

Deploy to Test

Auto Test

Deploy to Production

Measure/ Validate

Understanding the DevOps Pipeline

- The DevOps pipeline is a powerhouse for your IT lifecycle. It has the power to shorten your IT workflow, streamline its communication, add automation, and much more.

What is a DevOps Pipeline?

- A DevOps pipeline is a set of practices, tools, and automated processes used by development (Dev) and operations (Ops) teams to build and deploy code.
- Here's an overview of the two aspects of a software development lifecycle encompassing DevOps pipelines.

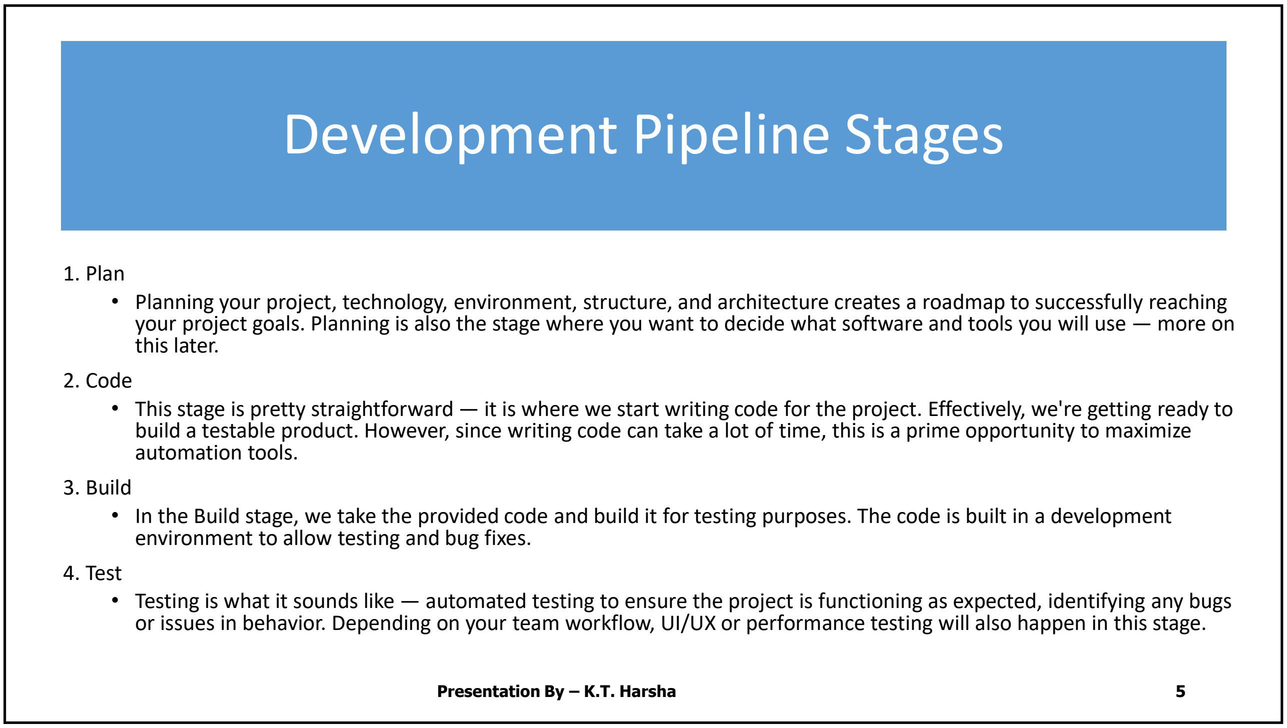
1. Development (Code Development)
   - The development phase consists of writing project code, testing, fixing bugs, building new features, conducting updates, and patching code.
   - The development phase has four steps: Plan, Code, Build, and Test. Here, developers use many tools to simplify the complexities of the development phase.
2. Operations (Deployment Operations)
   - The Operations phase is typically four steps: Release, Deploy, Operate, and Monitor. These steps can be just as complex as the development phase, with a lot going into seamlessly moving a software development project and any updates to production.

DevOps Pipeline Stages

- Let's now discuss the stages in the DevOps pipeline.
- We will break them into two phases — Dev and Ops — to help illuminate how they work together as a DevOps process. It is also important to mention that this entire process is very iterative, and you can expect to repeat these stages many times over.

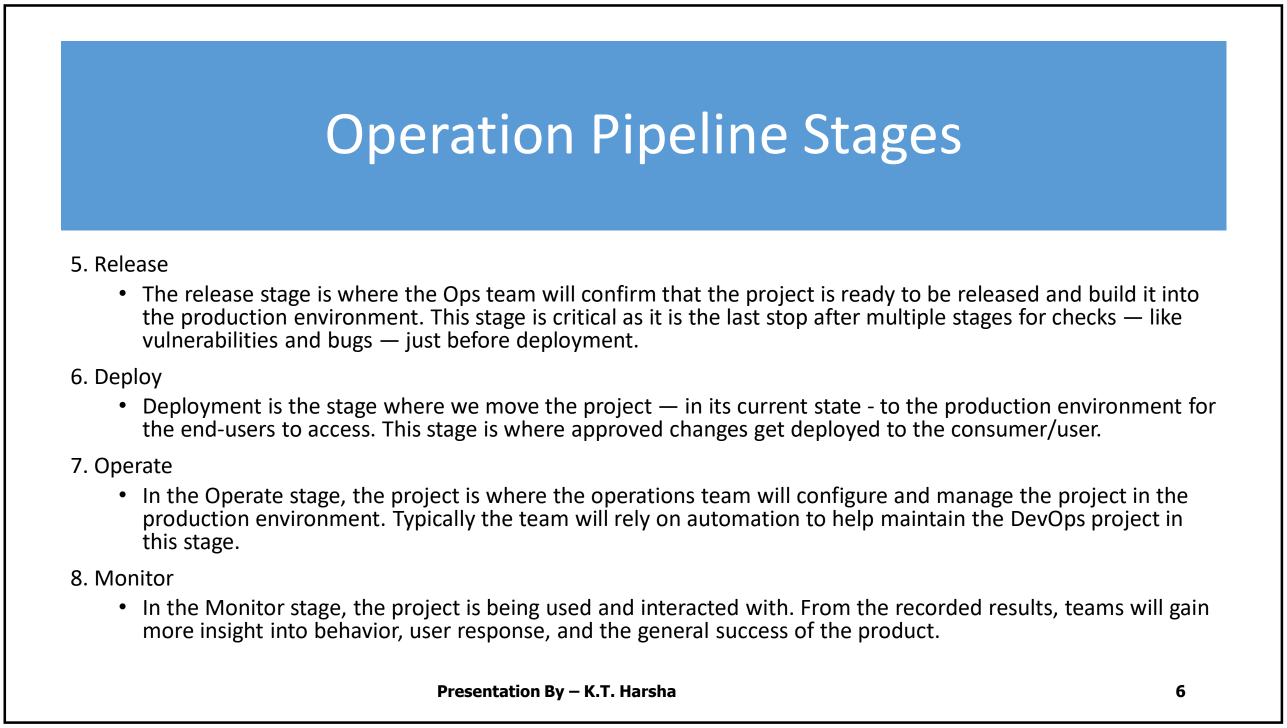
- Development Pipeline Stages
  1. Plan
  2. Code
  3. Build
  4. Test

- Operation Pipeline Stages
  5. Release
  6. Deploy
  7. Operate
  8. Monitor



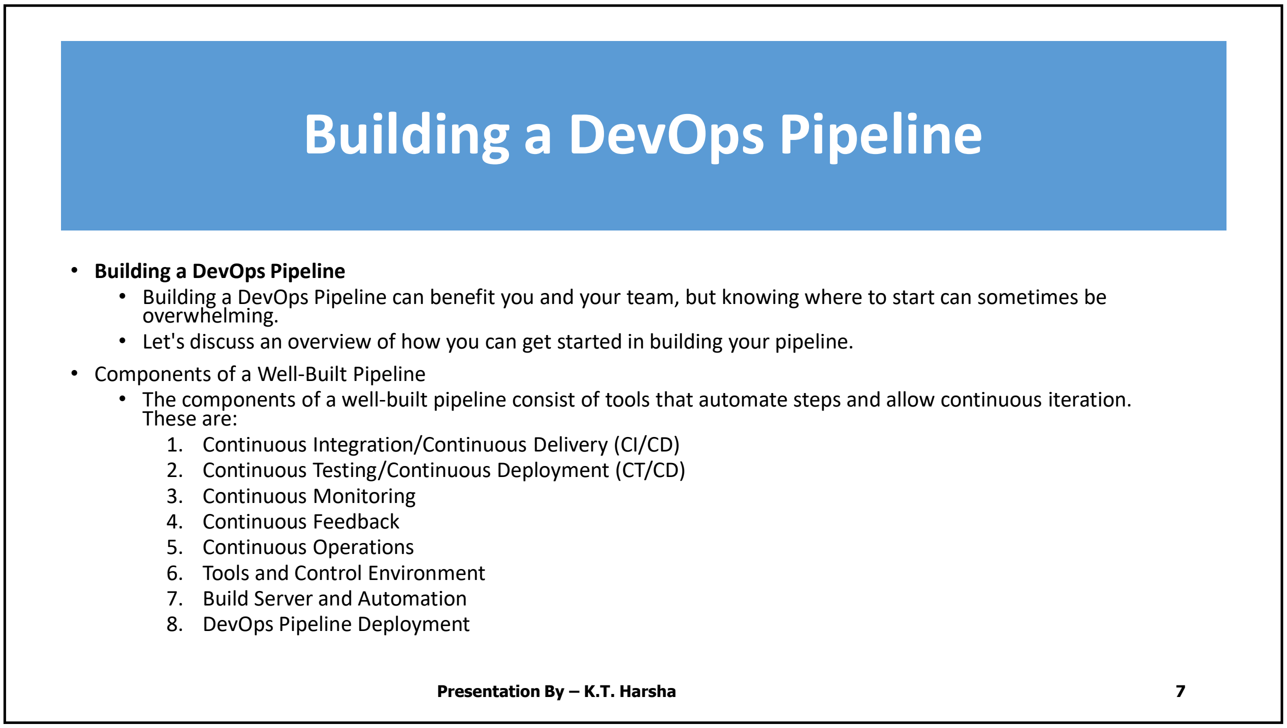
Development Pipeline Stages

1. Plan
   - Planning your project, technology, environment, structure, and architecture creates a roadmap to successfully reaching your project goals. Planning is also the stage where you want to decide what software and tools you will use — more on this later.
2. Code
   - This stage is pretty straightforward — it is where we start writing code for the project. Effectively, we're getting ready to build a testable product. However, since writing code can take a lot of time, this is a prime opportunity to maximize automation tools.
3. Build
   - In the Build stage, we take the provided code and build it for testing purposes. The code is built in a development environment to allow testing and bug fixes.
4. Test
   - Testing is what it sounds like — automated testing to ensure the project is functioning as expected, identifying any bugs or issues in behavior. Depending on your team workflow, UI/UX or performance testing will also happen in this stage.



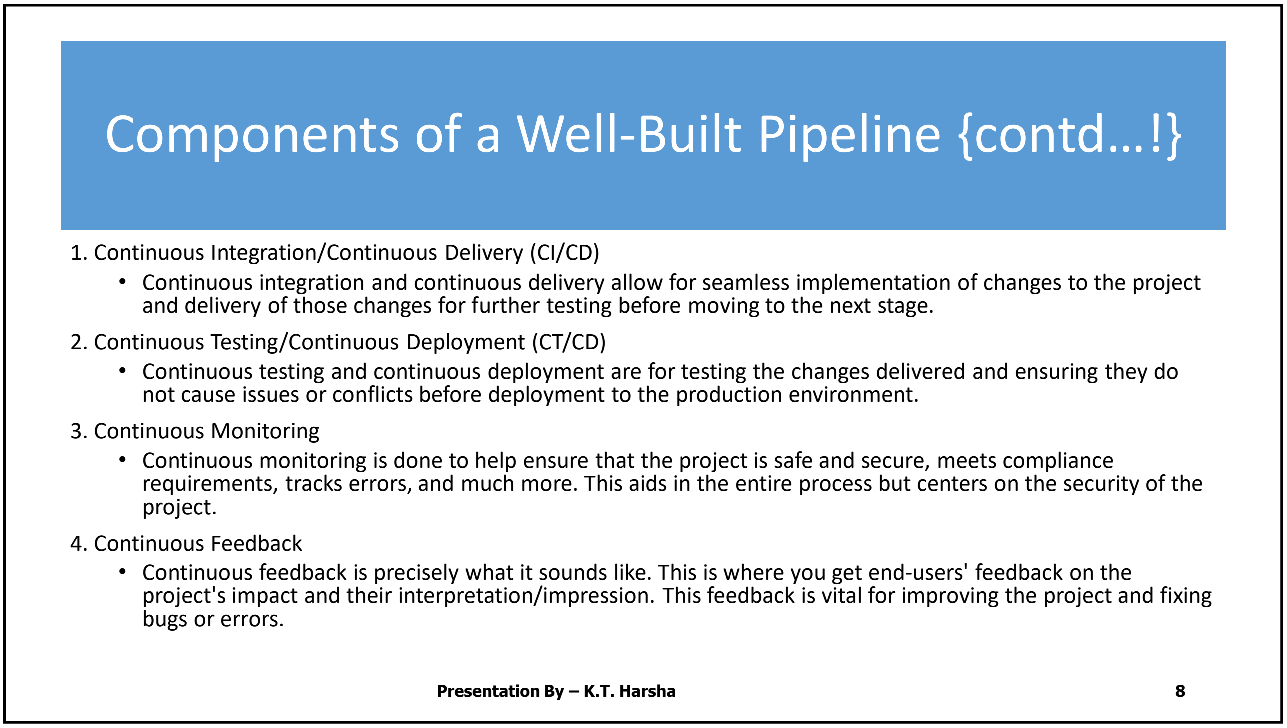
Operation Pipeline Stages

5. Release
   - The release stage is where the Ops team will confirm that the project is ready to be released and build it into the production environment. This stage is critical as it is the last stop after multiple stages for checks — like vulnerabilities and bugs — just before deployment.
6. Deploy
   - Deployment is the stage where we move the project — in its current state - to the production environment for the end-users to access. This stage is where approved changes get deployed to the consumer/user.
7. Operate
   - In the Operate stage, the project is where the operations team will configure and manage the project in the production environment. Typically the team will rely on automation to help maintain the DevOps project in this stage.
8. Monitor
   - In the Monitor stage, the project is being used and interacted with. From the recorded results, teams will gain more insight into behavior, user response, and the general success of the product.



Building a DevOps Pipeline

- **Building a DevOps Pipeline**
  - Building a DevOps Pipeline can benefit you and your team, but knowing where to start can sometimes be overwhelming.
  - Let's discuss an overview of how you can get started in building your pipeline.
- Components of a Well-Built Pipeline
  - The components of a well-built pipeline consist of tools that automate steps and allow continuous iteration. These are:
    1. Continuous Integration/Continuous Delivery (CI/CD)
    2. Continuous Testing/Continuous Deployment (CT/CD)
    3. Continuous Monitoring
    4. Continuous Feedback
    5. Continuous Operations
    6. Tools and Control Environment
    7. Build Server and Automation
    8. DevOps Pipeline Deployment



Components of a Well-Built Pipeline {contd…!}

1. Continuous Integration/Continuous Delivery (CI/CD)
   - Continuous integration and continuous delivery allow for seamless implementation of changes to the project and delivery of those changes for further testing before moving to the next stage.
2. Continuous Testing/Continuous Deployment (CT/CD)
   - Continuous testing and continuous deployment are for testing the changes delivered and ensuring they do not cause issues or conflicts before deployment to the production environment.
3. Continuous Monitoring
   - Continuous monitoring is done to help ensure that the project is safe and secure, meets compliance requirements, tracks errors, and much more. This aids in the entire process but centers on the security of the project.
4. Continuous Feedback
   - Continuous feedback is precisely what it sounds like. This is where you get end-users' feedback on the project's impact and their interpretation/impression. This feedback is vital for improving the project and fixing bugs or errors.

Components of a Well-Built Pipeline {contd…!}

5. Continuous Operations
    - This stage is for limiting planned downtime and preventing unplanned downtime. Operations will take steps to avoid issues and set up a plan of attack that leaves the project running, which helps maintain the pipeline flow — a very important step.
6. Tools and Control Environment
    - Researching tools and your preferred control environment for your DevOps pipeline is one of the most critical steps. These tools will effectively structure the rest of your DevOps process and workflow.
    - There are a lot of tools out there for the DevOps pipeline, and not every tool will meet your needs. Therefore, knowing which ones do and are compatible with each other is very important.
7. Build Server and Automation
    - The next step is to get your build server running to host your project and set up your security and data storage. This stage is also a prime opportunity to configure your automation technology and test it out in your new build server.
8. DevOps Pipeline Deployment
    - At this point, you should have everything set up and ready to go. The only thing left is deploying your DevOps pipeline and reaping the benefits.

How to Build a DevOps pipeline

- Now that you know the components of a well-built DevOps pipeline let’s consider how to build a typical pipeline.
    - ✓Step 1: Choose CI/CD Tool
    - ✓Step 2: Establish a Control Environment
    - ✓Step 3: Set up a build server
    - ✓Step 4. Initiate Automated Tests
    - ✓Step 5: Deploy to Production

How to Build a DevOps pipeline { Contd ..!}

Step 1: Choose CI/CD Tool

- Although DevOps methodologies might differ from company to company, each company must pick a CI/CD tool that suits its specific needs.
- The most common examples of these CI/CD tools are Jenkins, GitLab, Bamboo, TeamCity, and CirlceCi. Each tool has specific features and properties to help teams choose the best fit.

Step 2: Establish a Control Environment

- Whether your development team is small or not, you need a central location or place - known as a control environment — to store your source code.
- This control environment allows multiple developers to work together on the same codebase, avoid merge conflicts, and be on the same page regarding any changes made to your code.
- Excellent examples of source control management tools include Git, GitLab, and BitBucket.

How to Build a DevOps pipeline { Contd ..!}

Step 3: Set up a build server

- A build server or continuous integration (CI) server offers developers a centralized and reliable environment to build distributed development projects.
- Without a build server, it would be difficult to determine if the code works and is ready for production.

Step 4. Initiate Automated Tests

- After setting up your code on the build server, the next step is to test it.
- You can run automated unit, regression, functional, and integration tests, ensuring your code is error-free as you move on to the deployment stage.

How to Build a DevOps pipeline { Contd ..!}

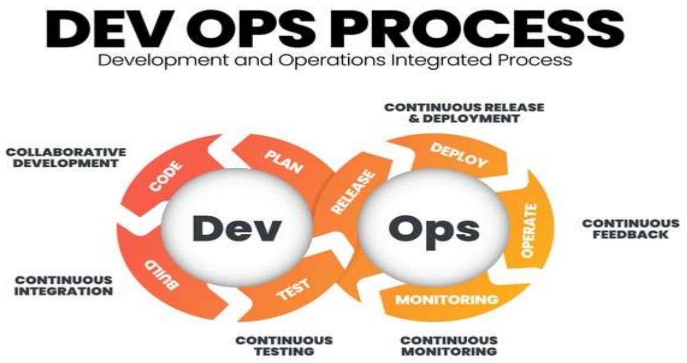
Step 5: Deploy to Production

- Here at the final stage, you can choose to deploy your code either manually or automatically.
- Deploying manually first is often recommended because you'd be able to monitor and quickly flag any issues that might come up. If no issues rears its head, you can then automatically deploy the code to save time and effort.
- You need a server infrastructure if you're deploying your code manually or automatically. The server infrastructure you use usually depends on the function you want it to perform and the type of software you want to deploy.

The Output of a DevOps Pipeline

- The DevOps pipeline is an iterative and largely automated process by design. In this vein, it utilizes variables to help maintain its various interdependent states to create the final project.
- The DevOps Pipeline has many tasks in its stages, and for each task, a variable is created to store and share data across the pipeline — such as permissions, users, and more. Sharing data through the pipeline makes for a robust lifecycle capable of scaling and adapting as needed.

DevOps Pipeline Example



- The best way to think of the DevOps pipeline is to imagine an assembly line for your project. At each assembly line step, things are added, removed, and tested to ensure a quality product.
- The DevOps pipeline works in the same way. Let's look at a visual example of this process.
- You may notice an overall concept across the pipeline — everything is continuous. These stages support a continuous process that should move fluidly throughout the entire DevOps pipeline.

Final Summary

- The DevOps pipeline is a series of eight stages in a project lifecycle that combine the functions of both teams - Development and Operations.
- The DevOps pipeline typically has eight stages.
  - In the Development phase, they are: plan, code, build, and test.
  - In the Operations phase, the stages are: release, deploy, operate, and monitor.
- The output of a DevOps pipeline is a collection of variables with assigned values used across the pipeline to pass data and manage project and user states.

Thank You