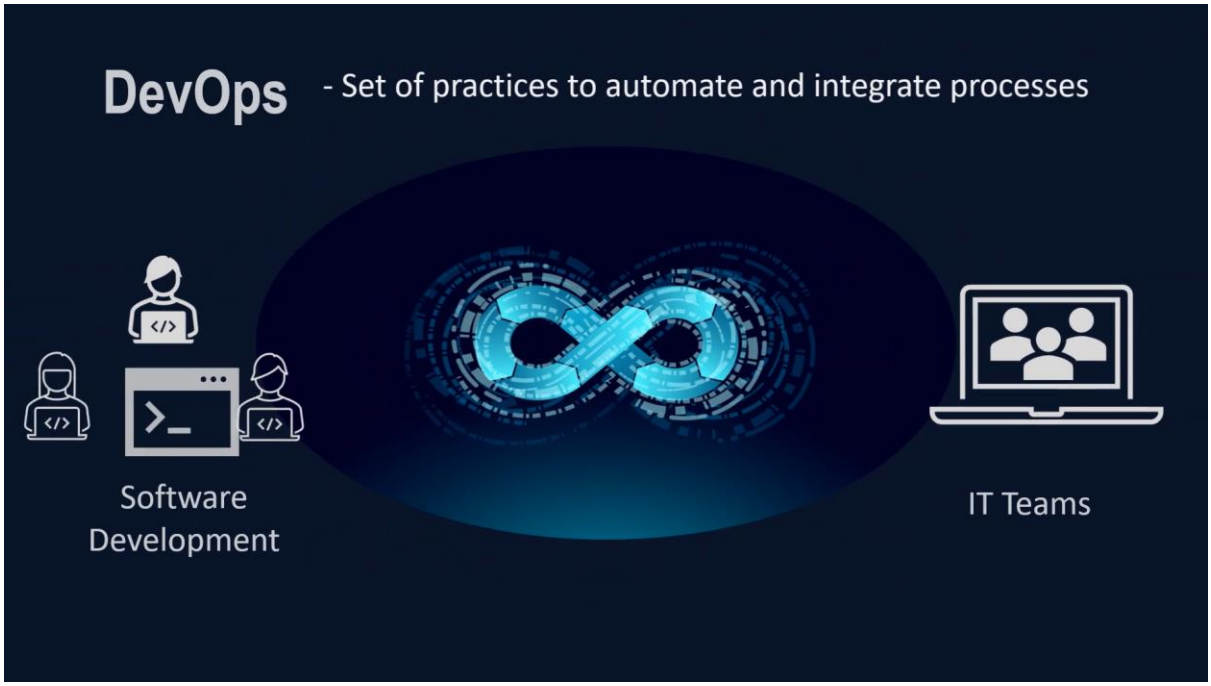
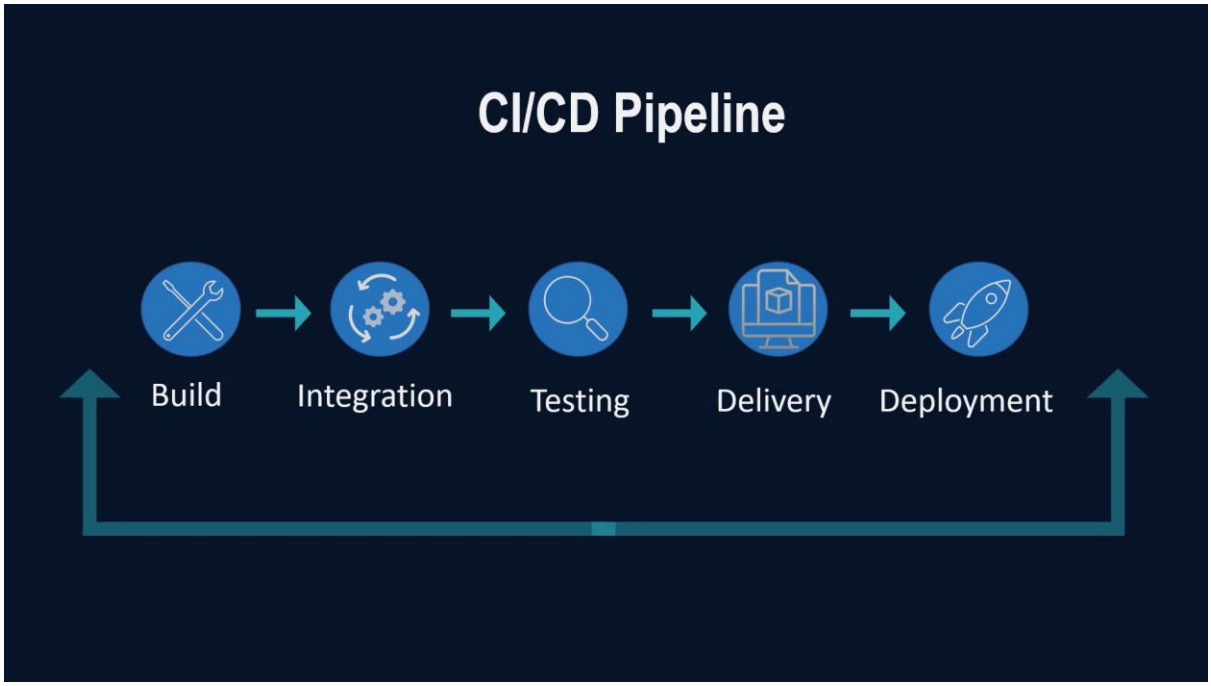

DevOps
- Set of practices to automate and integrate processes
Software Development
IT Teams


DevOps
Test
Build
Release
Release software in a fast and reliable manner

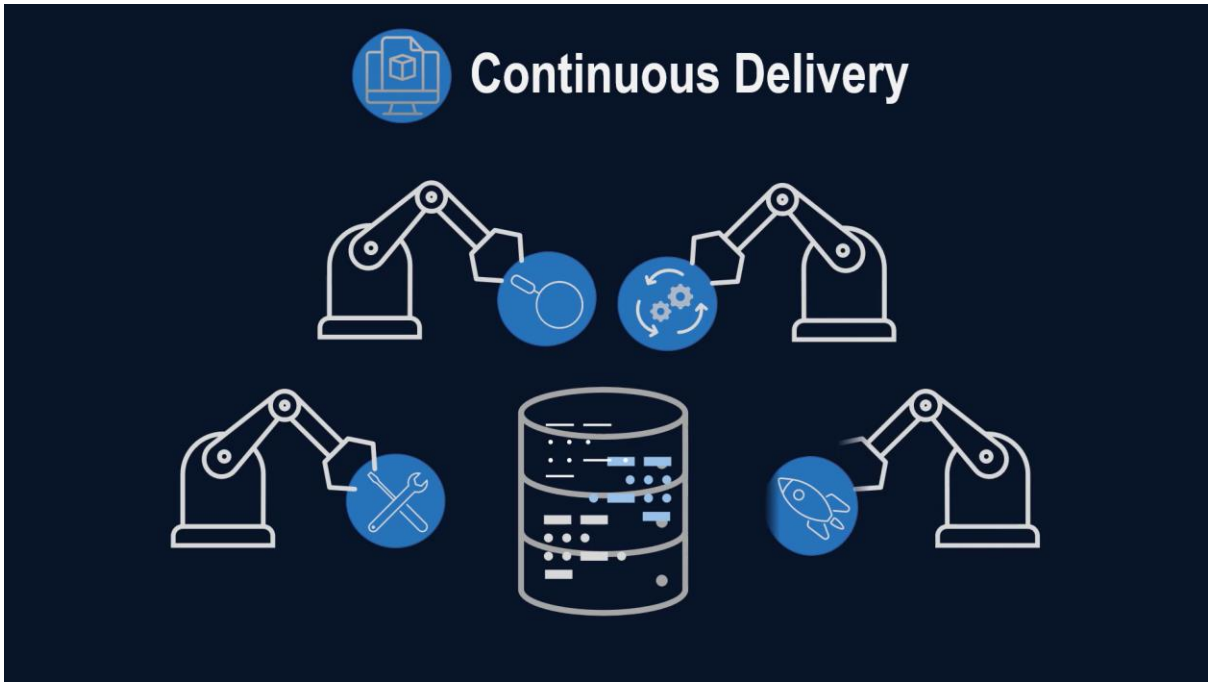

CI/CD Pipeline
Build
Integration
Testing
Delivery
Deployment


Continuous Delivery and Deployment
Build Once,
Deploy Anywhere

Continuous Integration

Continuous Integration

Automated Builds & Tests



Continuous Deployment

Continuous Deployment

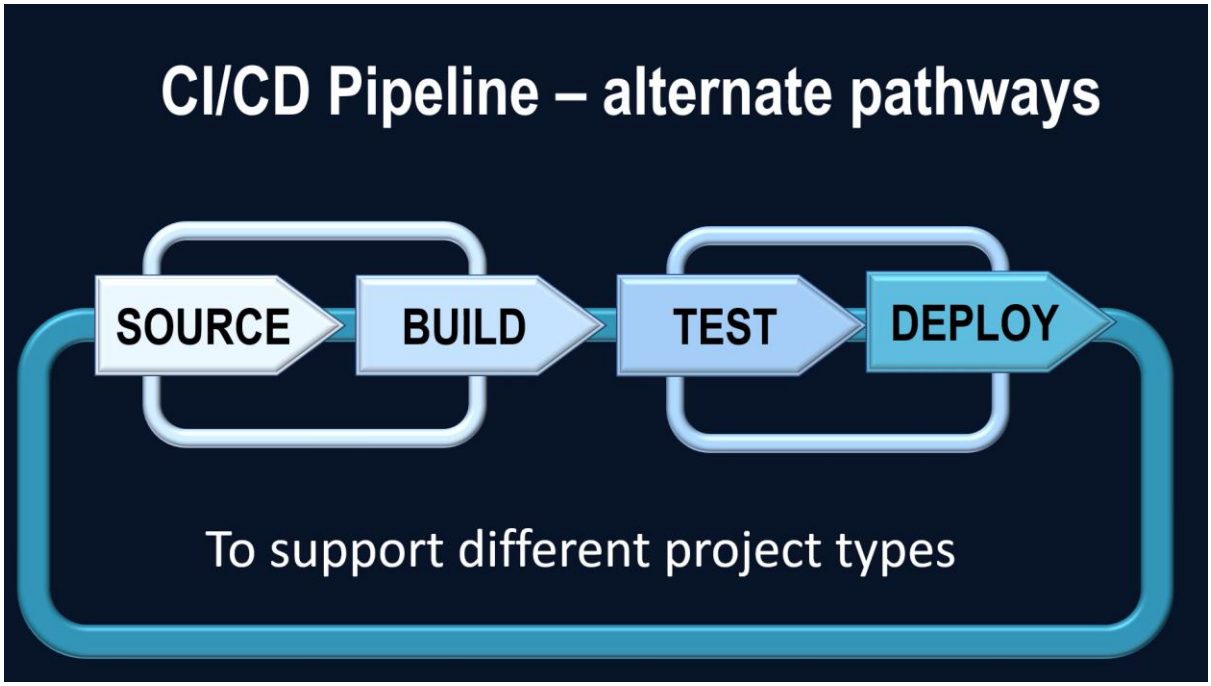
Automatically deploys each validated build to production

Continuous Deployment

Automatically deploys each validated build to production

Failed test will prevent a new change from being deployed

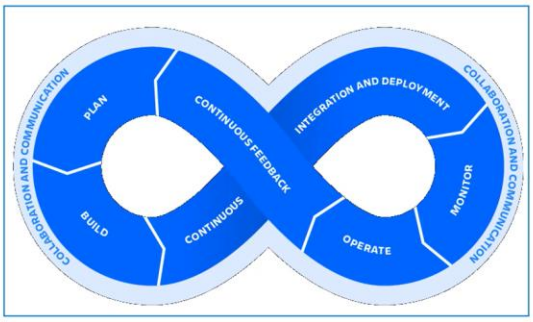
CI/CD Pipeline



WHAT IS DEVOPS?

TEKNOTURF

- DevOps
  - It is a set of practices that work to automate and integrate the processes between software development and IT teams, so that they can build, test, and release software faster and more reliably
  - DevOps was formed by combining the words "development" and "operations"
  - It bridges the gap between development and operations teams



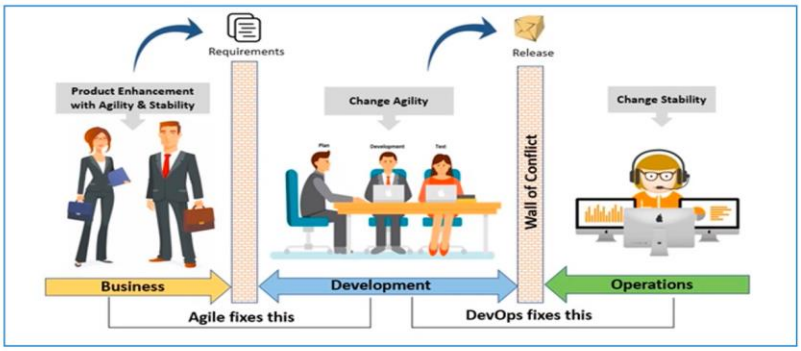
https://www.atlassian.com/devops

NEED FOR DEVOPS

- Initially a software is developed, tested and then sent to the operations team
- Before DevOps, there is a significant delay between development and operations
- Until any bugs are reported by the operations department or end users, the developers have to wait in order to rectify it
- Developers valuable time is delayed and hence progress for projects is also slow
- It is where DevOps came into the picture
- With DevOps, developer need not wait for the feedback, notifications will be sent to developers for their committed code
- Development and Operations teams work in collaboration to minimize the effort and risk involved in releasing software
- DevOps tool can easily point out the code which leads to failure during the build

DEVELOPMENT AND OPERATIONS

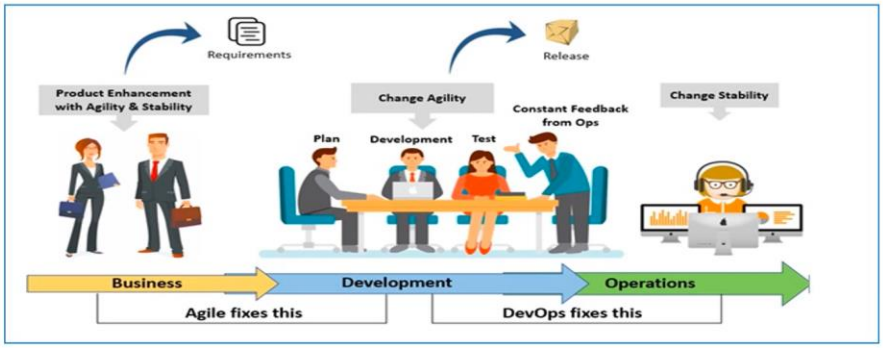
- Without DevOps



https://www.accenture.com/us-en/blogs/software-engineering-blog/shinde-development-operations-silos

DEVELOPMENT AND OPERATIONS (CONTINUED …)

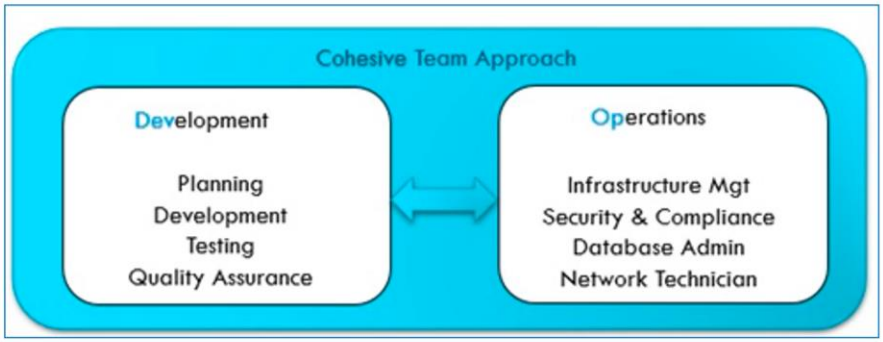
- With DevOps



https://www.accenture.com/us-en/blogs/software-engineering-blog/shinde-development-operations-silos

DEVELOPMENT AND OPERATIONS (CONTINUED …)

- Cohesive Teams



https://www.accenture.com/us-en/blogs/software-engineering-blog/shinde-development-operations-silos

DEVELOPMENT AND OPERATIONS (CONTINUED …)

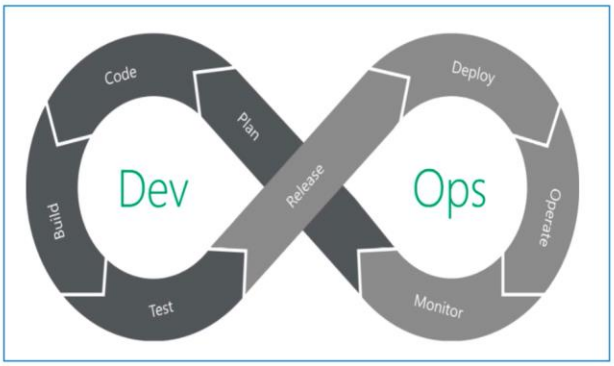
- Shared Objectives
  - Dev and Ops work cohesively toward common objectives
  - This also involves establishing collective team ownership
- Co-location
  - Co-location helps in improving collaboration between team members easily

DEVOPS PRINCIPLES

- Incremental Release
- Automation
- DevOps Pipeline
- Continuous Integration
- Continuous Delivery
- Continuous Monitoring
- Feedback sharing
- Version Control
- Collaboration

DEVOPS PHASES

- Plan
- Code
- Build
- Test
- Release
- Deploy
- Operate
- Monitor



https://medium.com/taptuit/the-eight-phases-of-a-devops-pipeline-fda53ec9bba

DEVOPS PHASES (CONTINUED …)

- Plan
  - It covers everything that happens before the developers start writing code
  - Requirements and feedback are gathered from stakeholders and customers
  - Product roadmap is built for future development
- Code
  - Team has a standard set of plug-ins installed in their development environments to aid the development process
  - Consistent code-styling is enforced to avoid security flaws
  - These tools also help to resolve issues that may fail tests later in the pipeline, resulting in fewer failed builds

DEVOPS PHASES (CONTINUED …)

- Build
  - Once a developer has finished a task, they commit their code to a shared code repository
  - Developer submits a pull request to merge their new code with the shared codebase
  - Another developer then reviews the changes and approve the pull-request
  - By continuously checking code changes into a shared repository and running builds and tests, we can highlight breaking bugs early in the development lifecycle
- Test
  - Once a build succeeds, it is automatically deployed to a staging environment for out-of-band testing
  - Staging environment may be an existing hosting service, or it could be a new environment provisioned as part of the deployment process
  - This practice of automatically provisioning a new environment at the time of deployment is referred to as Infrastructure-as-Code (IaC)
  - IaC is a core part of many DevOps pipelines
  - Once the application is deployed to the test environment, a series of manual and automated tests are performed

DEVOPS PHASES (CONTINUED …)

- Release
  - Release phase is a milestone in a DevOps pipeline
  - It's the point at which we say a build is ready for deployment into the production environment
  - Depending on the DevOps maturity of an organization, they may choose to automatically deploy any build that makes it to this stage of the pipeline
  - An organisation may want to have control over when builds are released to production
  - They may want to have a regular release schedule or only release new features once a milestone is met
- Deploy
  - Finally, a build is ready for the big time and it is released into production
  - Same Infrastructure-as-Code that built the test environment can be configured to build the production environment
  - A blue-green deployment lets us switch to the new production environment with no outage
  - If at any point, an issue is found with the new build, you can simply tell the hosting service to point requests back to the old environment

DEVOPS PHASES (CONTINUED …)

- Operate
  - New release is now live and being used by the customers
  - Operations team ensures that everything is running smoothly
  - Organization has also built a way for their customers to provide feedback on their service
  - This feedback loop is important
  - You need to capture this information
- Monitor
  - Final phase of the DevOps cycle is to monitor the environment
  - This builds on the customer feedback provided in the Operate phase
  - We can also do some introspection and monitor the DevOps pipeline itself
  - All of this information is then fed back to the Product Manager and the development team to close the loop on the process

CHARACTERISTICS OF DEVOPS

- Respect the organization's culture
- Take small steps
- Use system orchestration to get the benefits of automation
- Accommodate legacy systems wherever necessary
- Adopt a DevOps toolkit and then do it yourself

CHARACTERISTICS OF DEVOPS (CONTINUED …)

- Respect the organization's culture
  - Collaborative and respectful culture must be created across the company's entire IT organization with Development (Dev) and Operations (Ops) teams
  - Fundamental philosophy of DevOps is that developers, operations staff, and support teams must actively seek to work together
- Take small steps
  - Moving to a DevOps organization is easier if it is implemented through small steps with simpler and more frequent deployments, rather than one large change

CHARACTERISTICS OF DEVOPS (CONTINUED …)

- Use system orchestration to get the benefits of automation
  - Multi-domain system orchestration enables services to be managed and manipulated from end-to-end, from a central point and at a high level of abstraction
- Accommodate legacy systems wherever necessary
  - Large enterprises often have complex legacy infrastructure constraints
  - Implementing DevOps on one business practice might impact another application
  - Enterprises like these must balance the need to maintain legacy processes in some areas, while automating wherever possible

CHARACTERISTICS OF DEVOPS (CONTINUED …)

- Adopt a DevOps toolkit and then do it yourself
  - DevOps toolkit that an enterprise chooses must develop new, virtualized services quickly, customize them and differentiate them
  - Toolkit must allow enterprises to either build services themselves or utilize a professional services team to do it
  - A DevOps software offering choice, openness, empowerment, and self-service must be combined with a development approach
  - This model will allow enterprises to integrate their product development, IT and operations to create an interconnected ecosystem

BENEFITS OF DEVOPS

- Faster delivery time
- High collaboration between teams
- Greater customer experience
- Early defect detection
- Continuous release and deployment
- Innovative mindset

BENEFITS OF DEVOPS (CONTINUED …)

- Faster delivery time
  - DevOps utilizes automation to ensure a smooth flow of the Software Development Life Cycle(SDLC)
  - By promoting a collaborative culture, it offers the scope for quick and continuous feedback so that any glitches are fixed in time and the releases are done faster
- High collaboration between teams
  - DevOps provides mutual collaboration, communication, and integration across globally distributed teams in an IT organization
  - All team members, together, are responsible for meeting the quality and timelines of deliverables

BENEFITS OF DEVOPS (CONTINUED …)

- Greater customer experience
  - By automating the delivery pipeline, it becomes possible to ensure the reliability and stability of an application after every new release
  - When the applications perform flawlessly in production, organizations reap the benefit of greater customer satisfaction
- Early defect detection
  - Teams are empowered to share their feedback with each other so that the defects are detected as well as resolved early

BENEFITS OF DEVOPS (CONTINUED …)

- Continuous release and deployment
  - Today's software development practices require teams to continuously deliver quality software and to adapt shorter release cycles
  - DevOps enables this through automation
  - DevOps promotes better efficiency, higher quality, and faster & continuous releases
- Innovative mindset
  - In DevOps, deployment phases are more relaxed and there is immense scope for bringing an innovative approach for resolving issues

DEVOPS PRACTICES

DevOps Practices

- Continuous Integration
- Continuous Delivery
- Microservices
- Infrastructure as Code
- Monitoring And Logging
- Communication And Collaboration

DevOps Practices (Continued …)

- Continuous Integration
  - Continuous integration refers to the build and unit testing stages of the software release process
  - Every revision that is committed triggers an automated build and test

- Continuous Delivery
  - Continuous delivery automates the entire software release process
  - Every revision that is committed triggers an automated flow that builds, tests, and then stages the update
  - Final decision to deploy to a live production environment is triggered by the developer

DevOps Practices (Continued …)

- Microservices
  - Microservices architecture is a design approach to build a single application as a set of small services
  - Each service runs in its own process and communicates with other services through a well-defined interface
  - You can use different frameworks or programming languages to write microservices and to deploy them

DevOps Practices (Continued …)

- Infrastructure as Code
  - It is a practice in which infrastructure is provisioned and managed using code and software development techniques, such as version control and continuous integration
  - Cloud's API-driven model enables developers and system administrators to interact with infrastructure programmatically
  - Engineers can interface with infrastructure using code-based tools
  - Infrastructure and servers can quickly be deployed using standardized patterns
  - Configuration management
    - It frees developers and systems administrators from manually configuring operating systems, system applications, or server software
  - Policy as code
    - With infrastructure and its configuration coded with the cloud, organizations can monitor and enforce compliance dynamically and at scale
    - Infrastructure that is described by code can thus be tracked, validated, and reconfigured in an automated way

DevOps Practices (Continued …)

- Monitoring and Logging

- Organizations monitor metrics and logs to see how application and infrastructure performance impacts the experience of their product's end user
- Active monitoring becomes increasingly important as services must be available 24/7 and as application and infrastructure update frequency increases
- Communication and Collaboration
- Use of DevOps tooling and automation of the software delivery process establishes collaboration by physically bringing together the workflows and responsibilities of development and operations
- Teams set strong cultural norms around information sharing and facilitates communication through the use of chat applications, issue or project tracking systems, and wikis

CI/CD Concept

CI/CD Concept

- Continuous Integration
- Continuous deployment
- Continuous Delivery

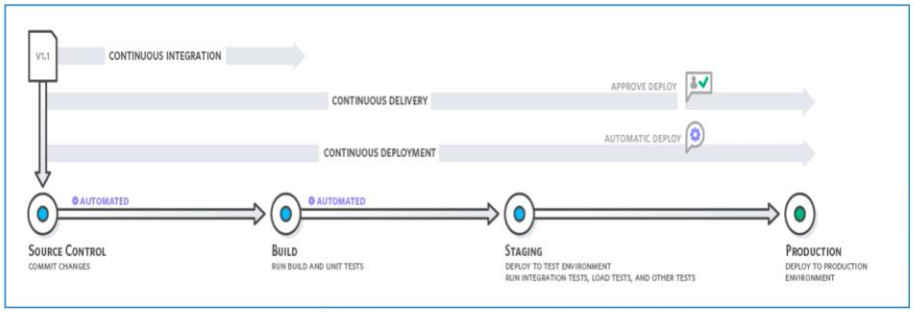
Continuous Integration (CI)

- Continuous Integration is a development practice where developers integrate code into a shared repository frequently, preferably several times a day
- Each integration can then be verified by an automated build and automated tests
- By working in small iterations, the software development process becomes predictable and reliable
- Developers can fix bugs quickly and usually discover them before they even reach users
- Results need to be transparently available to all team members
- Build status is reported to developers when they are changing the code

Need For Continuous Integration

- In the past, developers on a team might work in isolation for an extended period of time
- He/she merges their changes to the master branch once their work was completed
- This made merging code changes difficult and time-consuming
- By automating all integration steps, developers avoid repetitive work and human error
- CI tool monitors the central code repository and runs all automated tests on every commit

How does Continuous Integration Work?



https://aws.amazon.com/devops/continuous-integration/

Benefits Of Continuous Integration

- Improve Developer Productivity
- Find and Address Bugs Quicker
- Deliver Updates Faster

Continuous Delivery

- Continuous Delivery is an extension of continuous integration, since it automatically deploys all code changes to testing and/or production environment after the build stage
- Deploy to production as early as possible to make sure that you release small batches that are easy to troubleshoot in case of a problem

Benefits Of Continuous Delivery

- Automate the Software Release Process
- Improve Developer Productivity
- Find and Address Bugs Quicker
- Deliver Updates Faster

Continuous Deployment

- Continuous deployment goes one step further than continuous delivery
- Every change that passes all stages of the production pipeline is released to customers
- There's no human intervention, and only a failed test will prevent a new change to be deployed to production

Benefits of Continuous Deployment

- Reduced risk and cost
- Increased Innovation
- Enhanced Quality of release
- Increased customer feedback and satisfaction
- Enhanced visibility and tracking

Continuous Integration Vs. Continuous Delivery Vs. Continuous Deployment

Continuous Integration	Continuous Delivery	Continuous Deployment
CI is an approach of testing each change to codebase automatically.	CD is an approach to obtain changes of new features, configuration, and bug fixes.	CD is an approach to develop software in a short cycle.
CI refers to the versioning of source code.	CD refers to the logical evolution of CI.	CD refers to automated implementations of the source code.
CI focuses on automation testing to determine that the software has no errors or bugs.	Focuses on releasing new changes to your clients properly.	Emphasis on the change in all stages of your production pipeline.
CI is performed immediately after the developer checks-in.	In CD, developed code is continuously delivered until the programmer considers it is ready to ship.	In CD, developers deploy the c directly to the production stag when it is developed.
It helps you to identify and rectify issues early.	It allows developers to check software updates.	It enables you to rapidly deplo and validate new features and ideas.
It uses unit tests.	It uses business logic tests.	Any testing strategy is perform
Development team sends continuous code merging requests even when the testing process is running.	You deliver code for review that can be batched for release.	Deploy code using an automat process.
You require a continuous integration server to monitor the main repository.	You require a strong foundation in continuous integration.	You need a good testing culture.

CI/CD Tools

- CI/CD tools can help a team to automate their development, deployment, and testing
  - Some tools specifically handle the integration (CI) side
  - Some manage development and deployment (CD)
  - Others specialize in continuous testing or related functions
- Best known CI/CD tools:
  - Jenkins
  - CircleCI
  - TeamCity
  - Bamboo
  - GitLab

Jenkins

- It is the open source automation server that provides many plugins to support building, deploying and automating any project
- Jenkins is designed to handle anything from a simple CI server to a complete CD hub
- It is a server-based application and requires a web server like Apache Tomcat
- Since it monitors repeated tasks that arise during project development, it became popular
- CI/CD enables the development team to handle frequent code changes for faster and more reliable product delivery
- CI/CD pipeline is the set of steps aimed at improving the software delivery using the DevOps approach
- This entire process is automated using CI/CD tools like Jenkins

Why Use Continuous Integration With Jenkins?

Before Jenkins	After Jenkins
Once all Developers had completed their assigned coding tasks, they used to commit their code all at same time. Later, Build is tested and deployed.	The code is built and test as soon as the Developer commits code. Jenkins will build and test code many times during the day.
Since the code was built all at once, some developers would need to wait until other developers finish coding, to check their build	The code is built immediately after any of the Developer commits.
It is not an easy task to isolate, detect, and fix errors for multiple commits.	Since the code is built after each commit of a single developer, it is easy to detect whose code caused the built to fail.
Since code build and test process are entirely manual, there are a lot of chances for failure.	Automated build and test process saving timing and reducing defects.
The code is deployed once all the errors are fixed and tested.	The code is deployed after every successful build and test.
Development Cycle is slow.	The development cycle is fast. New features are more readily available to users. Increases profits.

Features Of Jenkins

- Continuous Integration and Continuous Delivery
- Distributed
- Easy configuration
- Easy installation
- Plugins
- Extensible

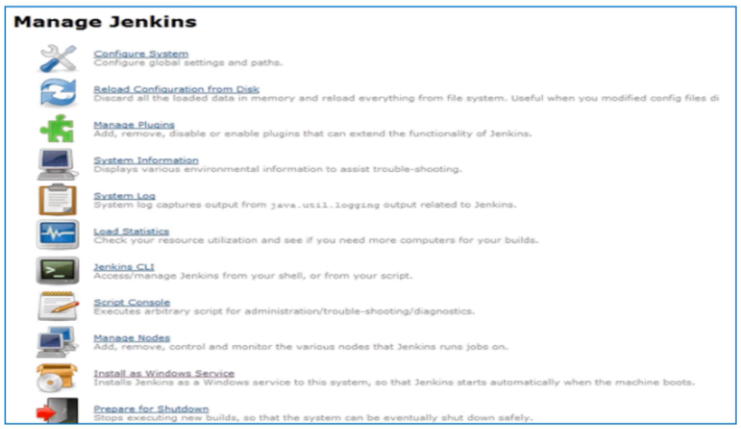
How Jenkins Work?

- Jenkins runs as a server on a variety of platforms
- It requires a Java 8 VM and above and can run on Oracle JRE or OpenJDK
- Jenkins runs as a Java servlet within a Jetty application server
- It can also run on other Java application servers such as Apache Tomcat
- Recently, Jenkins has been adapted to run in a Docker container
- To operate Jenkins, pipelines are created
- Pipelines are stored in a plain text Jenkinsfile
- Jenkins server then reads the Jenkinsfile and executes its commands
- This server also pushes the code down the pipeline from committed source code to production runtime

Jenkins Pipeline

- Jenkins Continuous Integration Pipeline consists of the following set of tools:
  - Continuous Integration Server
  - Source Control Tool
  - Build tool
  - Automation testing framework

Jenkins Plugins



Plugins Integration In Jenkins

CircleCI

- CircleCI's leading continuous integration and continuous delivery platform enables teams to deliver quality code, quickly
- It supports Windows, Linux, Docker, and macOS, and runs in the cloud or behind the firewall
- Teams can orchestrate complex workflows to move code through their ideal delivery pipelines
- It has an easily readable YAML configuration
- It does not require a dedicated server to run
- It is suitable for small projects

Why CircleCI?

- Clients tend to use CircleCI as their CI/CD tool, since jobs run faster and builds are well optimized
- CircleCI can be configured to run very complex pipelines efficiently
- It manages about one million tasks per day in aid to 30,000 organizations
- It further facilitates scalable performance-based pricing options
- You can SSH into any job to debug your build issues
- CircleCI can execute every task in a dedicated container or File

Features Of CircleCI

- Highly extensible with integrations
  - You can integrate CircleCI with AWS cloud ecosystem, Azure and several other environments/platforms to deploy the application
- Cloud, as well as Self-hosted CI/CD solution
  - CircleCI can be leveraged to orchestrate CI/CD pipeline:
    - In the CircleCI's cloud-hosted computes
    - Implement CI/CD by running CircleCI on your infrastructure
- Custom build environments & language support
  - CircleCI can be configured to imitate your target deployment environment by configuring your pipeline execution environment
  - It supports a wide range of languages, including PHP, Python, Java and so on
- Security and Performance
  - With CircleCI, you have application-level security and isolated runtime environment security
  - CircleCI is SOC II, and fedRamp certified
  - Effectively uses caching to speed-up builds

Working Of CircleCI

- CircleCI integrates with the code version control system
- Every time a developer pushes code to the repository, CircleCI creates a pipeline to process the code and test it
- CircleCI automatically runs this newly committed code in an isolated container or a virtual machine
- If the committed code successfully passes the pipeline, it gets pushed to the target staging or to the deployment environment

TeamCity

- TeamCity by JetBrains is a commercial, Java-based build management, and continuous integration server created by JetBrains
- It is known for its ease of setup, out-of-the-box usability, and beautiful user interface
- It can be installed on Windows and Linux servers
- It also provides support for the .Net framework and can be integrated into IDEs such as Visual Studio and Eclipse
- Conditional build steps are supported by this version
- You can also launch build agents in a Kubernetes cluster
- You can also integrate it with popular project management tools such as Azure DevOps and Jira

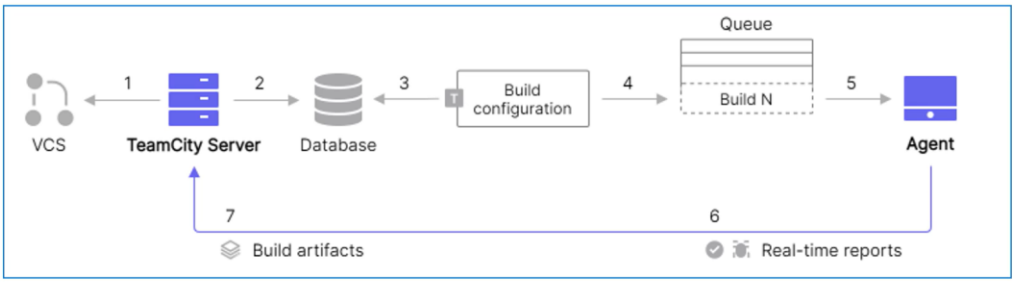
Why TeamCity?

- It offers an impressive set of features to match Jenkins
- It provides integration to Jira for tracking issues with ease
- It also integrates with IDEs such as Visual Studio and Eclipse
- It provides .NET framework support
- It has a build artifact repository integrated into it
- Free version of TeamCity - Professional server license provides 100 builds and three build agents at zero cost

Features Of TeamCity

- Technology Awareness
- Key Integrations
- Cloud Integrations
- Continuous Integration
- Configuration
- Build History
- Build Infrastructure
- Code Quality Tracking
- Extensibility and Customization
- VCS Interoperability

How TeamCity Works?



BAMBOO

- Bamboo is a continuous integration server developed by Atlassian
- It can be used to automate the release management for a software application, creating a continuous delivery pipeline
- It has built-in Git Branching workflows and deployment projects as well as built-in integrations with other Atlassian software
- It is available for platforms such as Windows, Mac OS X, and Linux
- Bamboo uses the concepts of agents to perform all CI-CD tasks including build, test and deploy
- There are two types of agents:
  - Local agents
    - Run as part of the Bamboo server as its process
  - Remote agents
    - Run on any other servers and computers
- Deployment is quite easy as it provides the feature of selecting the deployment project and “configure” to accomplish it

WHY BAMBOO?

- Bamboo becomes stronger with its built-in integration with other Atlassian tools, such as Jira Software and Bitbucket Server
- You can get visibility over code changes, build results, and deployments details throughout the platforms
- Provides an automated and reliable build/test process, leaving you free to code more
- Bamboo has a much user-friendly approach with a neat and intuitive user interface

FEATURES OF BAMBOO

- Documentation
- Platform dependency
- Plugin Support
- Compatibility

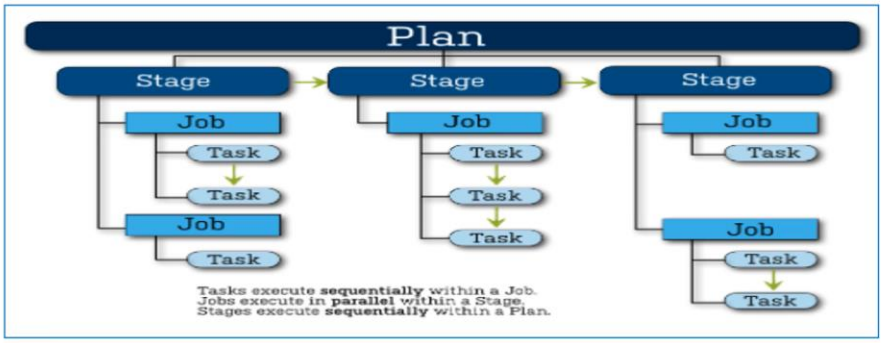
HOW BAMBOO WORKS?

TEKNOTURF

- Bamboo first gets your source from a source repository
- Bamboo starts the build
- Once your solution or project is built, you have "artifacts"
- You can do additional things with the build artifacts:
  - zip them up into a ZIP file and copy them somewhere
  - run an install builder on them and create an MSI
  - install them on a test server to make sure everything installs just fine
- Bamboo provides a web front-end for configuration for reporting the status of builds

HOW IS BAMBOO WORKFLOW ORGANIZED?

TEKNOTURF



https://confluence.atlassian.com/bamboo/understanding-the-bamboo-ci-server-289277285.html

GITLAB

- GitLab CI/CD was integrated into the main GitLab software with the GitLab 8.0 release
- GitLab CI/CD and GitLab are written in Ruby and Go and launched under an MIT license
- GitLab CI/CD is a UI based web application that manages projects or builds
- It is also accessible through APIs
- GitLab Runner processes your builds
- It runs the script by working in tandem with GitLab CI/CD through an API interface.
- Runners can run tests in any language across platforms
- A GitLab CI/CD server can manage more than 25,000 users

WHY GITLAB?

- Dev and Ops teams work together
- Only single application for the entire Devops cycle
- Security is built in
- Transparency is by default
- Everyone can contribute

FEATURES OF GITLAB

- High Availability Deployments
- Multi-environment, Multi-platform, Multi-language
- GitLab CI/CD Cloud vs. Self-hosted
- Integrations
- Security and performance
- Milestone Setting
- Code Reviews and Merge Requests
- Issue Tracking and Issue Shuffling
- Jekyll Plugins Support

HOW GITLAB WORKS?

- Install and register a runner for the project
- Define the CI/CD jobs in a structured order and with well-defined actions
- Save this YAML file as .gitlab-ci.yml file in the root directory of the project
- A GitLab CI/CD pipeline contains four stages
- If the code commit passes all these stages, it is successfully delivered/deployed

CICD AS A SERVICE

- Azure Pipelines
- AWS Pipelines
- Bitbucket Pipelines
- GitLab Pipelines

CI/CD PIPELINE

- A CI/CD pipeline is used to automate the process of continuous integration and continuous deployment
- Pipeline facilitates the software delivery process via stages like Build, Test, Merge and Deploy

STAGES OF CI/CD PIPELINE

- Source
- Build
- Test
- Deploy

AZURE DEVOPS

- Azure DevOps is a collection of services given by Microsoft Azure
- It provides development services for a team to support, plan, collaborate, build, and deploy applications
- It provides integrated features in a browser or an IDE(Integrated Development Environment)
- Some of the services for developers are :
  - Azure Repos
  - Azure Pipelines
  - Azure Boards
  - Azure Test Plans
  - Azure Artifacts
- These resources are quite useful, especially Azure Pipelines

AZURE PIPELINES

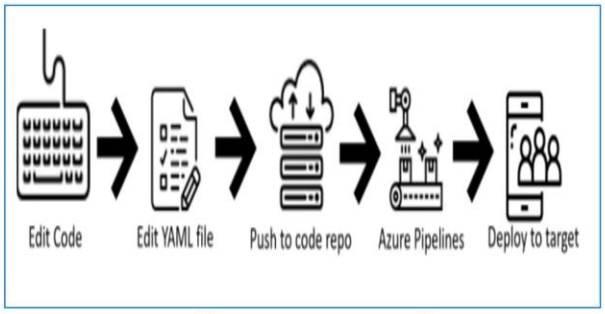
- Azure Pipelines supports continuous integration (CI) and continuous delivery (CD) to constantly and consistently test and build your code and ship it to any target
- You accomplish this by defining a pipeline
- You define pipelines using the YAML syntax or through the user interface(Classic)

ADVANTAGES OF AZURE PIPELINES

- Version Control Systems
- Programming Languages and Application Types
- Deployment Targets
- Pricing

DEFINE PIPELINES USING YAML SYNTAX

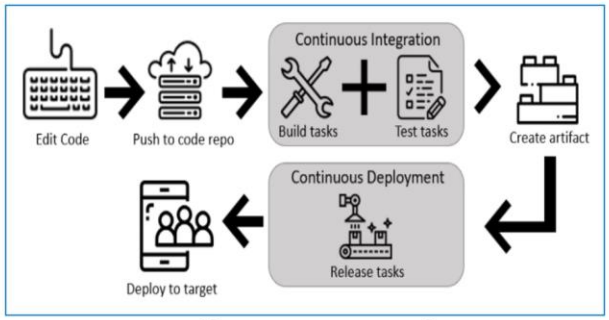
- You define your pipeline in a YAML file called azure-pipelines.yml with the rest of your app
- Pipeline is versioned with your code.
- It follows the same branching structure
- Every branch you use can modify the build policy by modifying the azure-pipelines.yml file
- A change to the build process might cause a break or result in an unexpected outcome
- Since the change is in version control with the rest of your codebase, you can more easily identify the issue



https://docs.microsoft.com/en-us/azure/devops/pipelines/get-started/pipelines-get-started?view=azure-devops

DEFINE PIPELINE USING CLASSIC INTERFACE

- Create and configure pipelines in the Azure DevOps web portal with the Classic user interface editor
- Define a build pipeline to build and test your code, and then to publish artifacts.
- Also define a release pipeline to consume and deploy those artifacts to deployment targets



https://docs.microsoft.com/en-us/azure/devops/pipelines/get-started/pipelines-get-started?view=azure-devops

HOW TO BUILD AZURE CI/CD PIPELINE?

- To create an Azure CI/CD Pipeline, you need to follow the below-given steps:
- Create an ASP.NET sample DevOps project using Azure DevOps Starter resource in Azure
- Examine Azure CI/CD pipelines configured by Azure DevOps Starter
- Clone the sample DevOps project to the system
- Commit the code and execute CI/CD
- Prerequisites
  - You will need an active Azure account for creating Azure Repos and Pipelines. You can create a Microsoft Azure Account for the same
  - For creating a sample app, we will be using Visual Studio and .NET Core. You can download Visual Studio and get .NET Core from the .NET Download Archives page

CICD AS A SERVICE

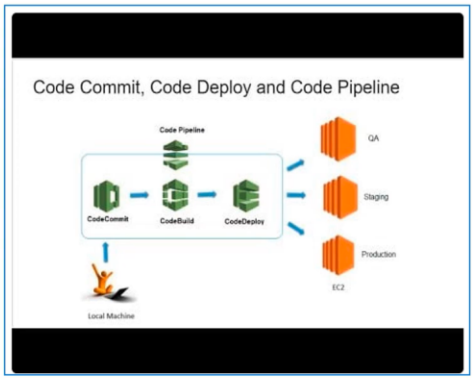
AWS CodePipeline

- It is a fully managed continuous delivery service that helps to automate release pipelines for fast and reliable application and infrastructure updates
- With this pipeline, you can model the full release process for building your code, deploying to pre-production environments, testing your application and releasing it to production
- Pipeline then builds, tests, and deploys the application according to the defined workflow whenever there is a code change
- You can integrate partner tools and own custom tools into any stage of the release process
- You only pay for what you use

https://aws.amazon.com/codepipeline/

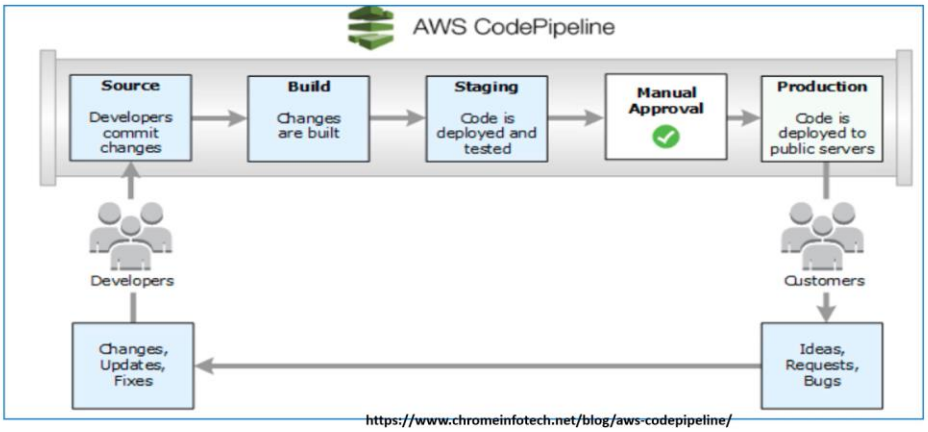
Why AWS CodePipeline?

- Enables to increase the speed and quality of software updates
- All the new changes run through a consistent set of Quality checks



https://dev.to/aws-builders/how-does-aws-codebuild-aws-codepipeline-compare-3e08

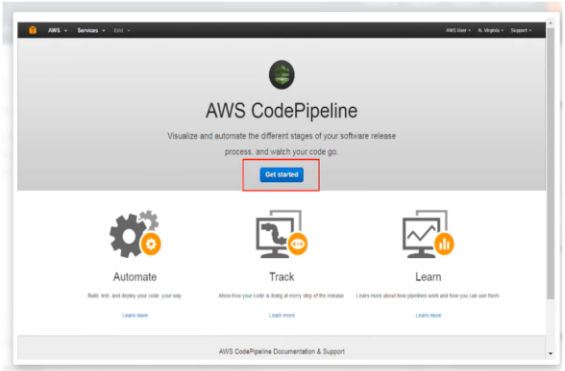
How AWSCodePipeline Work?



https://www.chromeinfotech.net/blog/aws-codepipeline/

Set Up A Continuous Deployment Pipeline

- Create the pipeline using AWS CodePipeline, a service that builds, tests, and deploys your code every time there is a code change
- Use your GitHub account, an Amazon Simple Storage Service (S3) bucket, or an AWS CodeCommit repository as the source location for the sample app's code
- Also use AWS Elastic Beanstalk as the deployment target for the sample app
- Completed pipeline will be able to detect changes made to the source repository containing the sample app and then automatically update the live sample app



https://aws.amazon.com/getting-started/hands-on/continuous-deployment-pipeline/

Set Up A Continuous Deployment Pipeline (Continued …)

- Step 1: Create a deployment environment
- Step 2: Get a copy of the sample code
- Step 3: Create the pipeline
- Step 4: Activate pipeline to deploy the code
- Step 5: Commit a change and then update your app
- Step 6: Clean up your resources

Bitbucket Pipelines

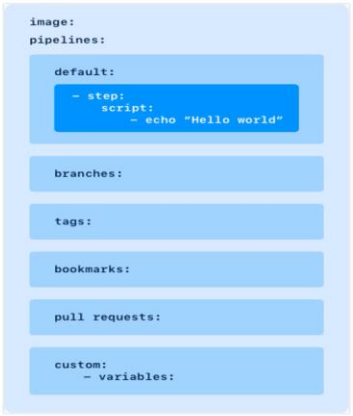
- It is an integrated CI/CD service built into Bitbucket
- It allows to automatically build, test, and deploy the code based on a configuration file in your repository
- Inside these containers, you can run commands
- It enables to build powerful, automated CI/CD pipelines

Features Of BitBucket

- Code Review
- Branch Permissions
- Pipelines
- Build integrations
- Jira Software Integration
- Trello boards
- Diff views
- Projects
- IDE integration

Understanding YAML File

- A pipeline is defined using a YAML file called bitbucket-pipelines.yml
- This file is located at the root of your repository



Pipes

- Pipes provide a simple way to configure a pipeline
- They are especially powerful when you want to work with third-party tools
- A pipe uses a script that lives in a Docker container
- Since the provided pipes are public, you can check the source code to see how it all works
- A pipe is made up of a few different files:
  - A script, or binary, is the code that performs the task
  - A Dockerfile, which tells us how to build the Docker container that runs your script
  - (Optional) metadata and readme docs, to make your pipe easy to understand
  - (Optional) some CI/CD configuration so that you can easily update the pipe

```
script:
  - pipe: atlassian/aws-s3-deploy:0.2.2
    variables:
      AWS_ACCESS_KEY_ID: $AWS_ACCESS_KEY_ID
      AWS_SECRET_ACCESS_KEY: $AWS_SECRET_ACCESS_KEY
      AWS_DEFAULT_REGION: 'us-east-1'
      S3_BUCKET: 'my-bucket-name'
      LOCAL_PATH: 'build'
```

How To Use Pipes?

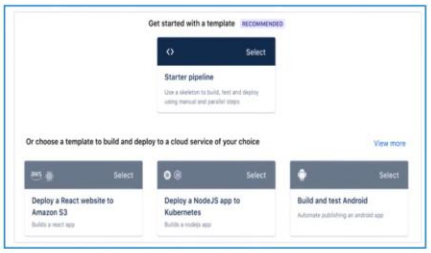
- There are two ways to add pipes to your pipeline:
  - Use the online editor
  - Edit the configuration directly

Configure Your First Pipeline

- There are two ways to configure your pipeline:
  - You can either directly write the YAML file
  - You can use the UI wizard provided by Bitbucket
- **Prerequisites**
  - You need to have a Bitbucket Cloud account
  - Your workspace must have at least one repository

Configure Your First Pipeline (Continued …)

- Steps for configuring the pipeline through UI wizard:
  - In Bitbucket, go to your repository and select Pipelines
  - Click "Create your first pipeline " option to scroll down to the template section
  - Choose one of the available templates. If you aren't sure, you can use the one RECOMMENDED
  - Once you choose a template, you will land in the YAML editor where you can configure your pipeline
  - Add Pipes
    - You can add pipes to easily configure a pipeline with third-party tools
  - Add variables
    - You can define custom variables that you can use in the YAML file
  - Outcome
    - You have configured the first pipeline



Available Templates

GitLab Pipelines

- GitLab is a fully integrated software development platform
- It is a lifecycle tool that provides a vast repository on web-based DevOps.
- It enables the team to be transparent, fast, effective, and cohesive from a discussion on a new idea to production, all on the same platform
- Code in Gitlab is available from many program languages and mostly it is written in Ruby and fewer parts are scripted in Go
- GitLab CI/CD is the part of GitLab that you use for all of the continuous methods
- With GitLab CI/CD, you can test, build, and publish your software with no third-party application or integration needed

Making Your First Project In GitLab

- Open gitlab.com
- Log in to your account
- After logging in, create a new project
- Choose a name, a description, and whether you want it to be private or publicly visible
- Open your Git Bash
- For the first step, configure your user name and email ID
- Then, create the first repository
- Navigate to this repository
- After navigating to this folder, you will find that the folder is currently empty
- Now it's time to initialize a git repository
- You will notice that something called the "master" appears on the screen. Whenever a Git repository is created for the first time, it creates a branch, and it's called the master
- Navigate to the folder, and find the hidden ".git" folder. This is created when a repository is initialized

Making Your First Project In GitLab (Continued …)

- Create a notepad file, type anything in it and save it
- Next step is to check the status of the file. The untracked files can be seen in red.
- Now, add the file to the staging area
- Next step is to commit the file
- Recheck the status of the file
- Now, it's time to push the notepad onto the GitLab repository
- Now go to your GitLab and check to see if there are any additions to the new project you initially created.
- You can see that the notepad appears there. You can now open and check the contents of the notepad

CI/CD Pipelines

- Pipelines are the top-level component of continuous integration, delivery, and deployment.
- Pipelines comprise:
  - Jobs, which define what to do
  - Stages, which define *when* to run the jobs
- Jobs are executed by runners
- In general, pipelines are executed automatically and require no intervention once created
- A typical pipeline might consist of four stages, executed in the following order:
  - A build stage, with a job called compile.
  - A test stage, with two jobs called test1 and test2.
  - A staging stage, with a job called deploy-to-stage.
  - A production stage, with a job called deploy-to-prod

GitLab CI/CD Process Overview

- To use GitLab CI/CD:
  - Ensure you have runners available to run your jobs
    - In GitLab, runners are agents that run your CI/CD jobs
    - You might already have runners available for your project
    - If no runners are listed on the Runners page in the UI, you or an administrator must install GitLab Runner and register at least one runner
  - Create a .gitlab-ci.yml file at the root of your repository
    - It is a YAML file where you configure specific instructions for GitLab CI/CD
  - View the status of your pipeline and job
    - When you committed your changes, a pipeline started.
    - A pipeline with three stages should be displayed

Set Up a Continuous Deployment Pipeline with GitLab CI/CD on Ubuntu 18.04

- Prerequisites:
  - Ubuntu 18.04 server
  - Docker installed on the server
  - A user account on a GitLab instance with an enabled container registry

Set Up a Continuous Deployment Pipeline with GitLab CI/CD on Ubuntu 18.04 (Continued …)

- Creating the GitLab Repository
- Registering a GitLab Runner
- Creating a Deployment User
- Setting Up an SSH Key
- Storing the Private Key in a GitLab CI/CD Variable
- Configuring the .gitlab-ci.yml File
- Validating the Deployment
- Rolling Back a Deployment