Introduction to DEV-OPS

By VIgnesh S

EVERYTHING STARTS WITH A IDEA OR A THOUGHT

IDEA..

OF
CREATING,
BUILDING,
DEVELOPING.

As a DEVELOPER,
You'll always create, build, develop a website, application, software or.. ANYTHING!

PROCESS

You’ll start by writing the CODE !

After completing the draft or the first version of the product you will SHARE it with others.

For that you have to HOST it in a server.

You need to CONFIGURE all the dependencies of the code, LIBRARIES, PACKAGES etc..

ENVIRONMENT

The Server or the ENVIRONMENT you used to RUN your application or HOSTING your website is ***PRODUCTION***

The Environment you used to write the code is ***DEVELOPMENT***

Additionally, We’ve to test the product. For this we’ll use the ***PRE-PRODUCTION*** environment

EXECUTION

The code would be written in a programming language as a TEXT-FILE.

To RUN it as an APPLICATION you need to convert it as an EXECUTABLE.

This Process is called as BUILDING. Maven, Gradle, Ant are popular tools used.

BUILD PROCESS

Source Code: Start with the source code, which is the human-readable code written by developers to create an application.

Compilation: In compiled languages like C++, the source code is transformed into machine code by a compiler.

In interpreted languages like Python, the code is executed directly, but it may be preprocessed or compiled to bytecode.

BUILD PROCESS

Build Automation: Use build automation tools like Make, Gradle, or Maven to manage and automate the build process.

These tools handle tasks such as compiling code, managing dependencies, and creating executable files or libraries.

Dependency Management: Ensure that all required libraries and dependencies are available.

Modern package managers like npm, pip, or Maven help download and manage dependencies automatically.

BUILD PROCESS

Testing: Run automated tests to check the code's correctness and functionality. This includes unit tests, integration tests, and sometimes end-to-end tests.

Artifact Generation: After successful testing, generate artifacts such as executables, binaries, or packages that can be deployed or distributed.

EXECUTION

Along with the process, The source code will be converted into a binary which includes the executable files. (./build.sh)

Then the binary is moved to production or pre-prod and then deployed. (./app)

DEVELOP - BUILD - DEPLOY !!

EXPLANATION

Will there be only a single developer working on the code ?

A company has multiple projects and hundreds of developers will be working on the code.

There will be a chaos in the code if its not managed properly.

GIT

Git is a distributed version control system widely used in the software development industry to track changes in source code and collaborate on projects.

GIT acts as a central REPOSITORY where you can PULL the latest code and make changes and push it back.

MANUAL PROCESS

A developers works on the DEV environment.

A dedicated build server is used to convert the code into an executable.

QA server is used for testing the product

And finally Deployed in PROD for user experience

CONS

All these are manual process and requires experience and skills.

Also there will be a lots of security updates, features and bug fixes are made to the code.

If the source code is updated again we should do the entire process right from the beginning.

CI / CD

To automate the manual process and release the features faster and more frequently,

CONTINUOUS INTEGRATION AND CONTINUOUS DEPLOYMENT tools are used.

Jenkins, Gitlab CI/CD, AWS codePipeline are some popular tools.

CI (Continuous Integration)

CI is a development practice where code changes are frequently and automatically tested and integrated into a shared repository.

It ensures that code changes don't break existing functionality and helps catch bugs early.

CD (Continuous Deployment/Delivery)

CD extends CI by automatically deploying code changes to production or staging environments after passing tests.

Continuous Deployment means automatically deploying every change to production, while Continuous Delivery involves automating the deployment but leaving the decision to deploy to production to a human.

Dependency Management Tools

Specific to your programming language or platform. These tools automate the process of installing and maintaining dependencies. For example:
Python: Use pip and requirements.txt for Python packages.

Node.js: Use npm or yarn and package.json for JavaScript packages.

Java: Use Maven or Gradle for Java dependencies.

STILL...

The developer will be very keen while handling the dependencies and packages throughout the entire process.

But we have ***developer(s)*** and also we have various environments to play with.

Installing and maintaining all the dependencies in all the environments is IMPOSSIBLE !!!

CONTAINERIZATION

Technology that allows you to package an application and all its dependencies, including libraries and configuration files, into a single, lightweight unit called a container.

Containers are isolated environments that can run consistently across different computing environments, such as development machines, test servers, and production servers.

CONTAINERIZATION

Developers creates a DOCKER FILE which specifies all the dependencies and creates an image along with the building process

By using a container IMAGE we can replicate the environment within seconds for running the application seamlessly.

This Solves the Problem !! Not Actually

CONS

What if you have 100 of containers running in 100 of instances ? Do you think you can manage it ?

While container images are indeed a powerful solution for creating consistent and reproducible application environments, they introduce a new challenge

Managing and coordinating multiple containers, especially in complex applications.

CONTAINER ORCHESTRATION

Container orchestration is a set of tools and practices for automating the deployment, scaling, management, and monitoring of containers in a clustered environment.

Container orchestration is a critical component of modern software development and deployment.

It helps address the challenges of deploying and running containerized applications at scale.

AGAIN ....

We’ve faced a lot more challenges as a DEV-OPS engineers. But Still we have a lot more problems to Solve……

Even though the lights of advance tools, containers and pods comes into play. Everything depends on the SERVERS.

Which requires Configuration and maintenance especially when you manage clusters.

WHY ?

SERVERS are the backbone of the entire process.

It should be provisioned and handled precisely.

The infrastructure and the architecture of the servers are very complicated to replicate along the clusters.

PROVISIONING TOOL

A software or set of tools used in IT and system administration to automate the process of setting up and configuring infrastructure resources, servers, and software applications.

The goal of provisioning tools is to streamline and standardize the deployment and management of computing resources.

Infrastructure Provisioning

Server Deployment: Provisioning tools automate the creation and configuration of servers, whether physical or virtual. They can provision servers in data centers, cloud environments, or on-premises infrastructure.

Networking: In addition to servers, provisioning tools often handle network configuration, including setting up firewalls, load balancers, and network policies.

Storage: Provisioning tools may also allocate and configure storage resources, such as disks and volumes.

Examples of Provisioning Tools

Terraform: Infrastructure-as-code tool that enables the provisioning of resources across various cloud providers and on-premises infrastructure.

AWS CloudFormation: Amazon Web Services (AWS) provides CloudFormation for provisioning and managing AWS resources using templates.

Azure Resource Manager (ARM): Microsoft Azure offers ARM templates to provision and manage Azure resources.

ANSIBLE

Ansible is a powerful automation and configuration management tool that simplifies the deployment, configuration, and management of infrastructure and applications.

It uses a declarative language to define the desired state of systems and then automates the process to achieve that state.

ANSIBLE

Imagine you have a lot of computers or servers, and you want to make sure they're all set up the same way, with the right software and settings. Thats where ansible comes into play.

So, why do we need Ansible? Because it saves us time and makes sure our computers work correctly. It's like having a helpful assistant who can set up and take care of all our computers,

OHH YES! IT’s DONE!!

But It’s NOT …

Yes we have developed, build and deployed the application by having all the superpowers in the world. We crafted everything with finesse !!

But TECH itself is EPHEMERAL !!

We can't be sure things stay normal!!

Server monitoring

Essential practice in the world of DevOps and system administration.

It involves the continuous observation and collection of data about a server's performance, health, and various metrics.

This information is crucial for ensuring the reliability, security, and optimal functioning of servers.

Importance of Server Monitoring

Proactive Issue Detection

Server monitoring allows you to detect potential issues and anomalies in real-time.

By continuously collecting data on server performance, you can identify irregularities before they turn into major problems.

Optimizing Resource Usage

Monitoring provides insights into resource utilization, such as CPU, memory, disk space. With this data, you can efficiently handle the usage of hardware and minimizing costs.

AND NOW IT’s Officially DONE

THANK YOU!!