AmazingDevOps

Welcome to Amazing DevOps

AmazingDevOps.com url: https://amazingdevops.com/ **Contact Information: +1 302 235 9992**

AmazingdevOps.Com
DevOps Curriculum

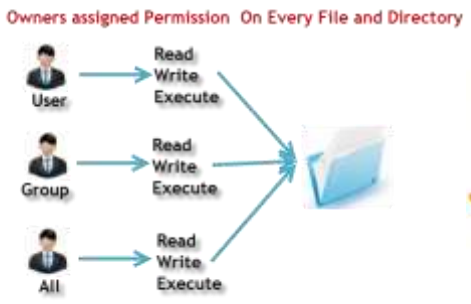
Month 1: Introduction to DevOps, Linux Fundamentals, and Tools

CODE
PLAN
RELEASE
DEPLOY
DEV
OPS
BUILD
TEST
MONITOR
OPERATE

1. **Introduction to DevOps:**
   - Understanding DevOps culture, principles, and practices
   - Benefits and challenges of DevOps
   - Role of DevOps in software development and delivery
   - Continuous Integration/Continuous Deployment (CI/CD) pipeline
   - Day-to-Day Activities of a DevOps Engineer
2. **Linux Fundamentals:**
   - Introduction to the Linux operating system
   - File system and directory structure
   - Basic command-line tools and utilities
   - Process management and monitoring
   - Text editors: Vim, Emacs, Nano, etc.
3. **Ubuntu and CentOS:**
   - Introduction to Ubuntu and CentOS Linux distributions
   - Installation and basic configuration
   - Package management: apt, yum, rpm
   - User and group management
   - Basic system administration tasks
4. **Git and GitHub:**
   - Introduction to version control systems (VCS)
   - Git fundamentals: repository, commit, branch, merge, conflict, pull, push.
   - GitHub: creating a repository, collaborating with others, pull requests, code reviews.

5. **Virtualization and Hypervisors:**
   - Introduction to virtualization concepts
   - Hypervisors: VMware, VirtualBox, KVM
   - Installation and configuration of VirtualBox
   - Creation of virtual machines, snapshots, cloning
6. **Vagrant:**
   - Introduction to Vagrant: what is it, why use it.
   - Installation and setup
   - Creation of Vagrant file
   - Provisioning with shell scripts: installing software, configuring services, networking.

Month 2: Files, File permissions, Bash Scripting, and Introduction to AWS Cloud



Files Permission:

- Understanding ownership and permissions of files in Unix-based systems
- Changing ownership and permissions of files using chmod and chown commands
- Using symbolic links to create shortcuts to files or directories.
- Compressing and decompressing files using gzip and tar commands.
- Archiving files and directories using tar command

Bash scripting:

- Understanding variables and their types in Bash
- Conditional statements in Bash (if/else, case statements)
- Loops in Bash (for, while, until loops)
- Functions in Bash (defining and calling functions)
- Error handling in Bash (return codes, exit status)
- Passing command-line arguments to Bash scripts

AWS Cloud:

- Introduction to AWS (history, services, benefits)
- Setting up an AWS account and enabling MFA (multi-factor authentication)
- Installing and configuring AWS CLI (Command Line Interface) for managing AWS services

- Understanding AWS IAM (Identity and Access Management) and creating IAM users and policies
- Creating and managing S3 (Simple Storage Service) buckets in AWS
- Uploading, downloading, and managing files in S3 buckets

Month 3: Terraform Cloud, AWS Cloud, AWSCLI, Python3.x, Boto3, Pip.

Terraform Cloud:

- Understanding what Terraform Cloud is and its benefits.
- Setting up an account on Terraform Cloud and configuring workspaces.
- Remote state management in Terraform Cloud

Terraform CLI:

- Installing Terraform CLI on your local server.
- Initializing a Terraform project.
- Planning and applying changes using Terraform CLI
- Destroying resources using Terraform CLI

AWS environmental variables:

- Understanding AWS environmental variables and their purpose
- Setting AWS environmental variables on your local machine using export
- Unsetting AWS environmental variables on your local machine

Setting up Terraform pipeline:

- Understanding the purpose of a Terraform pipeline.
- Setting up Git and GitHub for version control
- Integrating Terraform Cloud and AWS Cloud into your pipeline.
- Cloning a Git repository
- Initializing a Terraform project, planning changes, and applying changes.

HashiCorp Configuration Language (HCL):

- Understanding the syntax and structure of HCL
- Working with variables, data types, and operators in HCL
- Using functions and modules in HCL

Terraform configuration files:

- Understanding the purpose and structure of Terraform configuration files.
- Working with variables and outputs in Terraform configuration files.
- Configuring providers and resources in Terraform configuration files.
- Working with data sources and modules in Terraform configuration files.

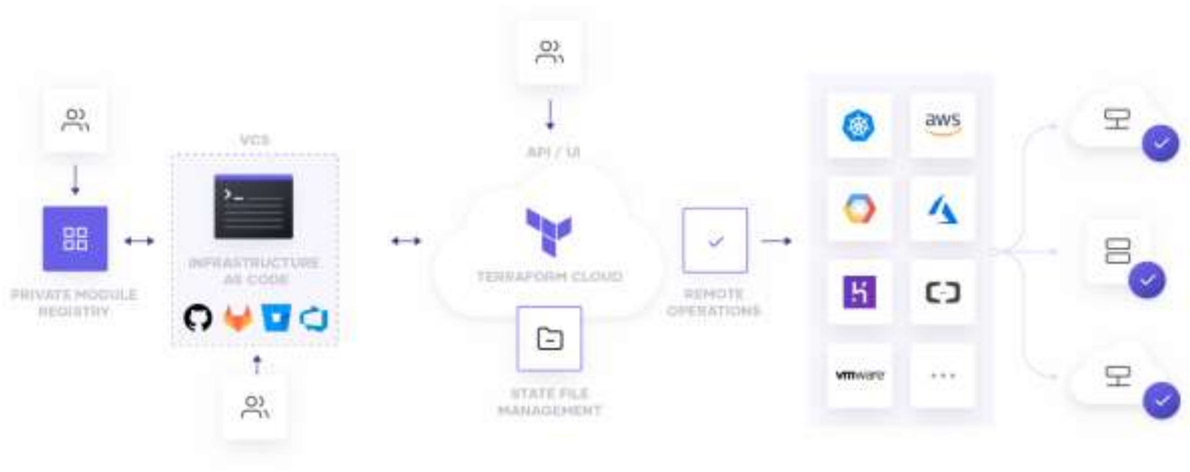
Python 3.x:

- Installing Python 3.x on your local machine.
- Understanding data types in Python.
- Working with control structures in Python (if/else, loops).
- Defining and calling functions in Python.
- Working with modules and error handling in Python.

Boto3:

- Installing Boto3, the AWS SDK for Python.
- Managing EC2 instances using Boto3.
- Managing S3 buckets using Boto3.

Month 4: terraform Automation, Terraform Code, Levels of terraform Code, and Python Automation



HashiCorp Configuration Language (HCL):

- Syntax: HCL is a declarative language used to write configuration files. The syntax is like JSON, but it's designed to be more human-readable and user-friendly.
- Variables: HCL supports variables, which can be used to store and reuse values within a configuration file. Variables are defined using the "variable" keyword and can be used in expressions throughout the file.
- Data types: HCL supports a range of data types, including strings, numbers, Booleans, lists, and maps. Each data type has its own syntax for defining and using values.
- Operators: HCL supports a range of operators, including arithmetic, comparison, and logical operators. These operators can be used in expressions to manipulate and compare values.

- Functions: HCL provides a range of built-in functions, including string manipulation, mathematical operations, and data transformation functions. These functions can be used to simplify complex expressions and calculations.
- Modules: HCL supports modularization, which allows you to break up a configuration file into smaller, reusable modules. Modules can be defined within the same file or in separate files and can be used to organize and simplify complex configurations.

Terraform configuration files: 7 levels.

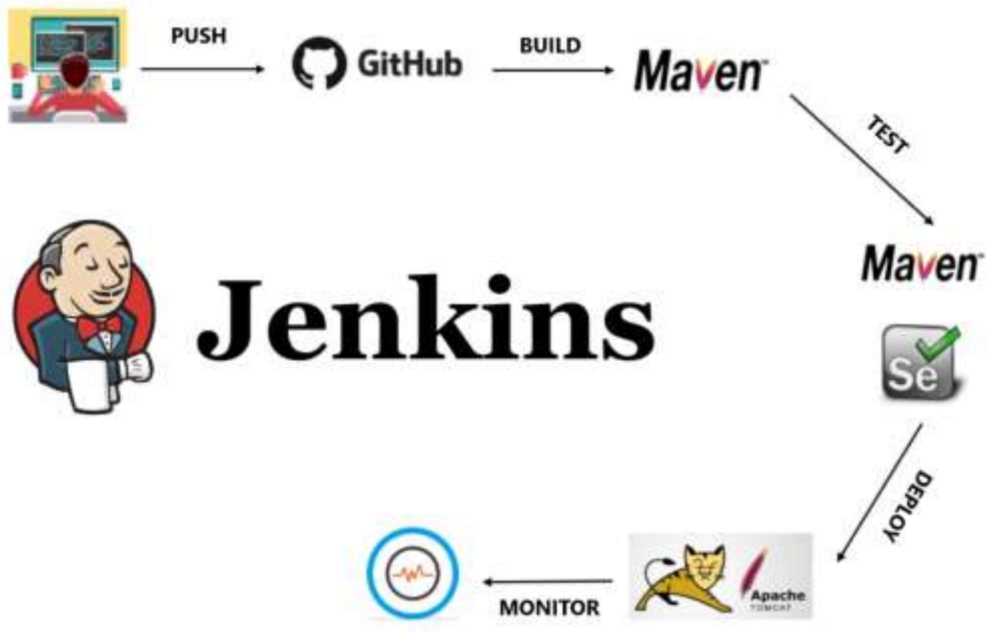
- Variables: Terraform supports variables, which can be used to store and reuse values within a configuration file. Variables are defined using the "variable" keyword and can be used in expressions throughout the file.
- Outputs: Terraform supports outputs, which allow you to export values from a configuration file for use in other files or scripts. Outputs are defined using the "output" keyword and can be used to share information between different parts of a Terraform configuration.
- Providers: Terraform uses providers to interact with different infrastructure platforms, such as AWS, Azure, or GCP. Providers are defined using the "provider" keyword and are used to configure the connection to a specific platform.
- Resources: Terraform uses resources to define the infrastructure components you want to create or manage. Resources are defined using the "resource" keyword and are used to specify the type of resource you want to create (e.g., an EC2 instance or a VPC).
- Data sources: Terraform uses data sources to retrieve information from an existing infrastructure component. Data sources are defined using the "data" keyword and are used to retrieve information about existing resources (e.g., the ID of a specific EC2 instance).
- Modules: Terraform supports modularization, which allows you to break up a configuration file into smaller, reusable modules. Modules can be defined within the same file or in separate files and can be used to organize and simplify complex configurations.

Python 3.x: Python Automation

- Installation: Python 3.x can be installed on a range of platforms, including Windows, MacOS, and Linux. The installation process may vary depending on your platform, but typically involves downloading and running an installer.
- Data types: Python supports a range of data types, including integers, floating-point numbers, strings, Booleans, lists, tuples, and dictionaries. Each data type has its own syntax for defining and using values.
- Control structures: Python supports a range of control structures, including if statements, for loops, while loops, and try-except blocks. These structures can be used to control the flow of a program and make decisions based on specific conditions.

- Functions: Python supports the creation of functions, which can be used to encapsulate and reuse blocks of code. Functions are defined using the "def" keyword and can accept input arguments and return output values.
- Modules: Python supports modularization, which allows you to break up a program into smaller, reusable modules. Modules can be defined within the same file or in separate files and can be used to organize and simplify complex programs.

Month 5: DevOps Concepts, Jenkins CI/CD Pipeline, Docker, Containers and Microservices



DevOps concepts:

- Continuous Integration (CI)
- Continuous Delivery (CD)
- Continuous Deployment (CD)
- Infrastructure as Code (IaC)
- Configuration Management
- Monitoring and Logging

Docker:

- Docker containers
- Docker images
- Docker Hub
- Dockerfile

- Docker Compose

Microservices:

- Microservices architecture
- Service discovery
- API gateway
- Load balancing
- Scalability
- Fault tolerance

Jenkins server:

- Installation
- Configuration
- Security
- Plugins
- Jenkinsfile

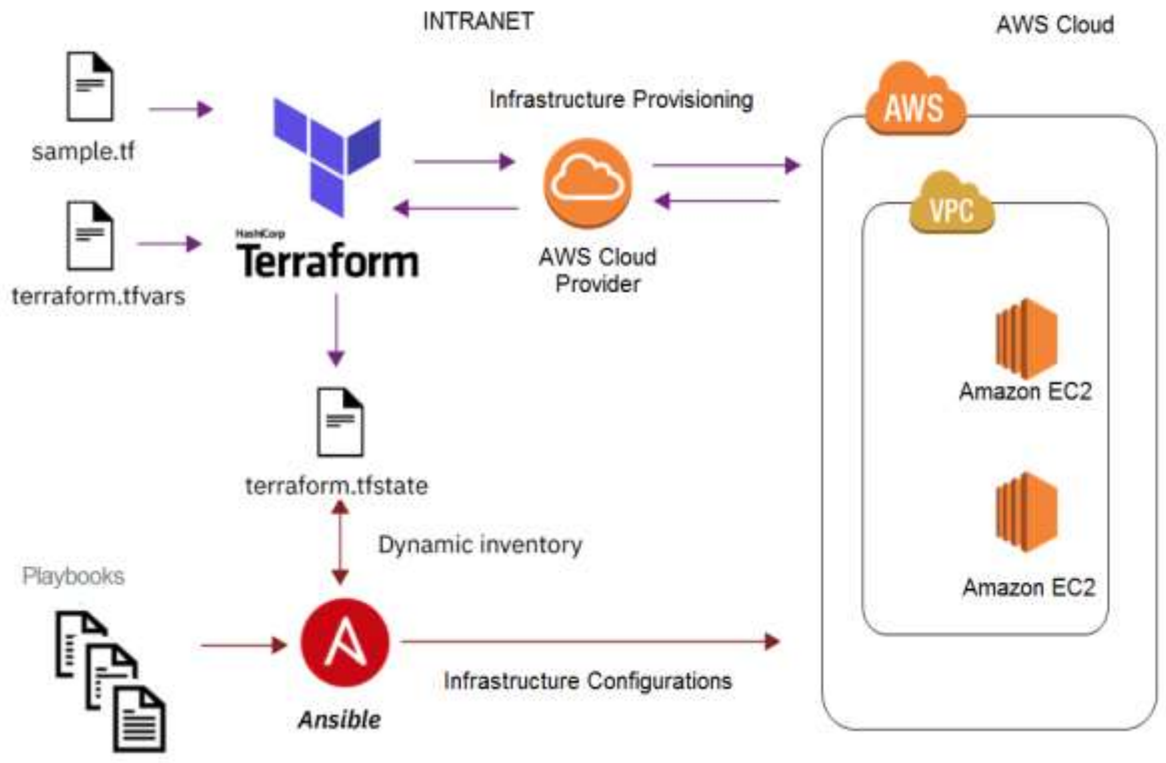
Jenkins CI/CD Automation Pipeline:

- Stages
- Steps
- Triggers
- Artifacts
- Notifications
- Parallel execution

Jenkins script:

- Declarative and Scripted
- Groovy syntax
- Variables
- Functions
- Conditional logic

Month 6: Ansible, Google Kubernetes Cluster, Kubernetes Declarative tools, CloudWatch & Alarms, Prometheus, Grafana and ELK



Ansible:

- Introduction
- Installation
- Inventory
- Modules
- Playbooks
- Roles
- Templates

Ansible Tower:

- Web-based GUI
- Job scheduling
- REST API
- Workflows
- Notifications
- Access control

Ansible Automation:

- Use cases.
- Best practices
- Testing
- Debugging
- Reporting

Google Kubernetes Engine (GKE):

- Introduction
- Cluster creation
- Node pool management
- Workload deployment

Deploying Microservices to GKE:

- FluxCd
- ArgoCd
- Progressive Deployment with Flagger
- Istio mesh
- NGINX Ingress Controller
- Istio Service Mesh

Monitoring:

- CloudWatch
- CloudWatch Alarms
- Prometheus
- Grafana
- ELK (Elasticsearch, Fluentd, Kibana)

A Day: Resume preparation and job support

- Career advice
- Interview preparation

- Resume writing
- Job search strategies
- Job application support as a DevOps Engineer or Cloud Architecture or Cloud Engineer

Our Location

3616 Kirkwood Hwy, Wilmington, DE 19808, United States

+1 302 235 9992

itsupport@amazingdevops.com

https://amazingdevops.com/