| **Name : Parth Kadam** | **Class/Roll No. :D11AD/27** | **Grade :** |
| --- | --- | --- |

**Title of Experiment :** To understand DevOps: Principles, Practices, and DevOps Engineer Role and Responsibilities

**Objective of Experiment :** To understand DevOps practices which aims to simplify Software Development Life Cycle.

**Outcome of Experiment :** To understand the fundamentals of DevOps engineering and be fully proficient with DevOps terminologies, concepts, benefits, and deployment options to meet your business requirements

**Theory :**

DevOps takes two previously separated teams – software development and IT operations – and turns them into one united front that creates secure code while speeding up the software development lifecycle. DevOps fundamentals include a collaborative and communicative culture, automated testing, releases and deployments, and frequent iteration. Another commonly used term in the DevOps space is DevSecOps, which refers to a DevOps practice with a specific emphasis on security.

DevOps principles:

Automation of the Software Development Lifecycle:

DevOps automates processes like continuous integration, delivery, and infrastructure management, ensuring quick and consistent validation of code changes.

Collaboration and Communication:

DevOps promotes collaboration among cross-functional teams, breaking down silos, and aligning goals to improve communication and shared responsibility.

Continuous Improvement and Minimization of Waste:

DevOps emphasizes iterative development, feedback loops, and root cause analysis, fostering a culture of continuous learning, improvement, and waste reduction.

Hyperfocus on User Needs with Short Feedback Loops:

DevOps accelerates deployment with features like continuous deployment, feature toggles, and monitoring, ensuring quick user feedback and a user-centric development approach.

DevOps Practices:

Continuous Integration

Continuous integration is a software development practice where developers regularly merge their code changes into a central repository, after which automated builds and tests are run. The key goals of continuous integration are to find and address bugs quicker, improve software quality, and reduce the time it takes to validate and release new software updates.

Continuous Delivery

Continuous delivery is a software development practice where code changes are automatically built, tested, and prepared for a release to production. It expands upon continuous integration by deploying all code changes to a testing environment and/or a production environment after the build stage. When continuous delivery is implemented properly, developers will always have a deployment-ready build artifact that has passed through a standardized test process.

Microservices

The 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 using a lightweight mechanism, typically an HTTP-based application programming interface (API). Microservices are built around business capabilities; each service is scoped to a single purpose. You can use different frameworks or programming languages to write microservices and deploy them independently, as a single service, or as a group of services.

Infrastructure as Code

Infrastructure as code is a practice in which infrastructure is provisioned and managed using code and software development techniques, such as version control and continuous integration. The cloud’s API-driven model enables developers and system administrators to interact with infrastructure programmatically, and at scale, instead of needing to manually set up and configure resources. Thus, engineers can interface with infrastructure using code-based tools and treat infrastructure in a manner similar to how they treat application code. Because they are defined by code, infrastructure and servers can quickly be deployed using standardized patterns, updated with the latest patches and versions, or duplicated in repeatable ways.

Configuration Management

Developers and system administrators use code to automate operating system and host configuration, operational tasks, and more. The use of code makes configuration changes repeatable and standardized. It frees developers and systems administrators from manually configuring operating systems, system applications, or server software.

Policy as Code

With infrastructure and its configuration codified 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. This makes it easier for organizations to govern changes over resources and ensure that security measures are properly enforced in a distributed manner (e.g. information security or compliance with PCI-DSS or HIPAA). This allows teams within an organization to move at higher velocity since non-compliant resources can be automatically flagged for further investigation or even automatically brought back into compliance.

Monitoring and Logging

Organizations monitor metrics and logs to see how application and infrastructure performance impacts the experience of their product’s end user. By capturing, categorizing, and then analyzing data and logs generated by applications and infrastructure, organizations understand how changes or updates impact users, shedding insights into the root causes of problems or unexpected changes. Active monitoring becomes increasingly important as services must be available 24/7 and as application and infrastructure update frequency increases. Creating alerts or performing real-time analysis of this data also helps organizations more proactively monitor their services.

Communication and Collaboration

Increased communication and collaboration in an organization is one of the key cultural aspects of DevOps. The 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. Building on top of that, these teams set strong cultural norms around information sharing and facilitating communication through the use of chat applications, issue or project tracking systems, and wikis. This helps speed up communication across developers, operations, and even other teams like marketing or sales, allowing all parts of the organization to align more closely on goals and projects.

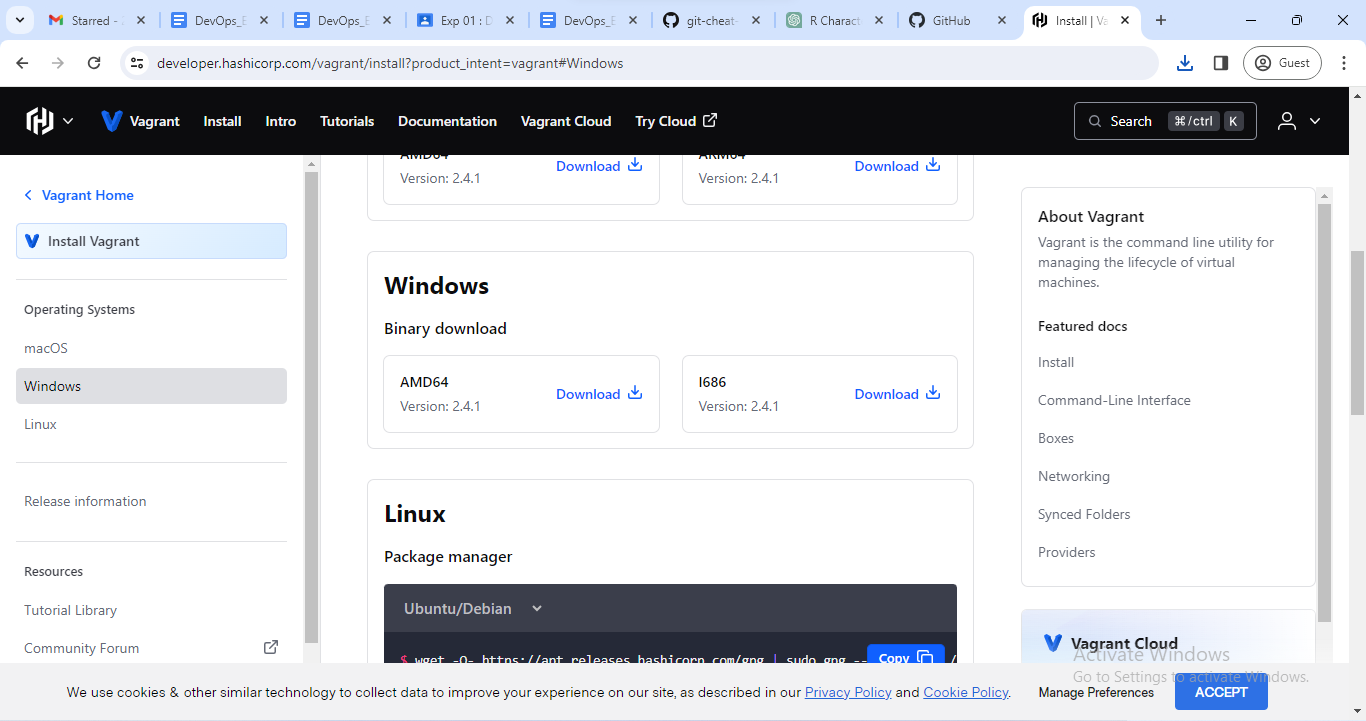
A DevOps Engineer is responsible for automating and optimizing software development processes, fostering collaboration between development and operations teams, implementing CI/CD pipelines, managing infrastructure as code, ensuring security, monitoring system performance, and continuously improving processes. The role requires expertise in automation tools, cloud platforms, and a commitment to enhancing the efficiency and reliability of software delivery.

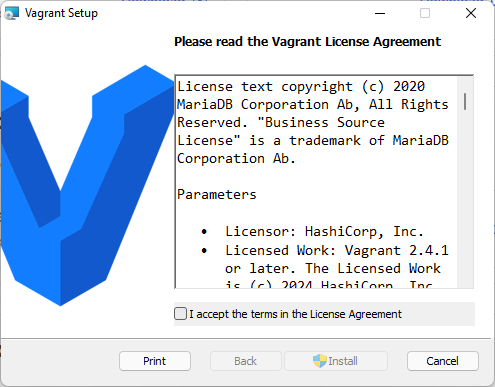
**Vagrant**

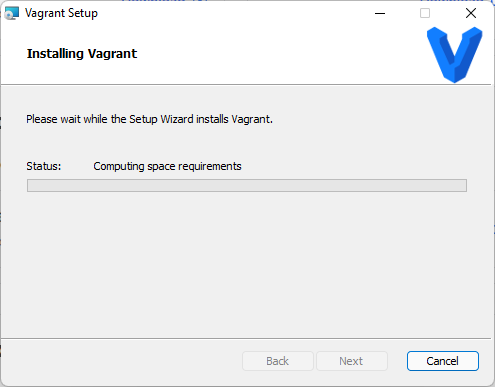
Vagrant is a tool for building complete development environments. With an easy-to-use workflow and focus on automation, Vagrant lowers development environment setup time, increases development/production parity, and makes the "it works on my machine" excuse a relic of the past.

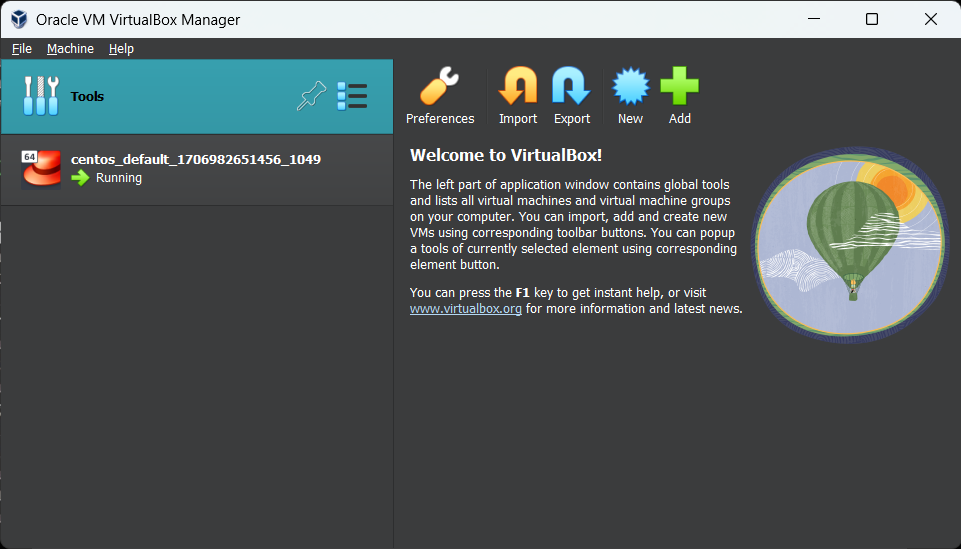
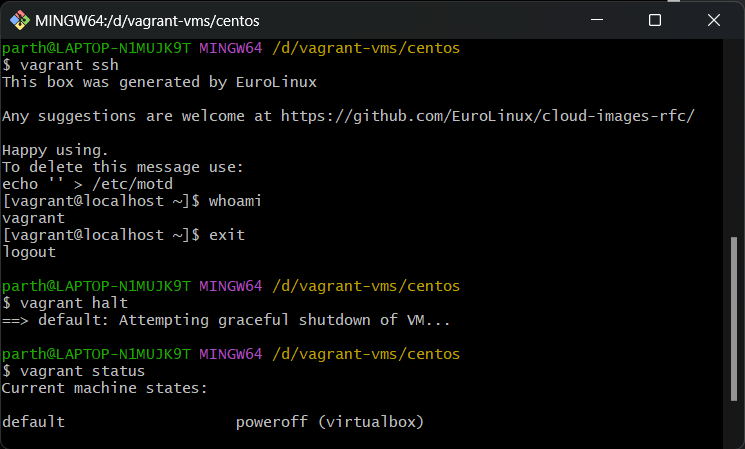
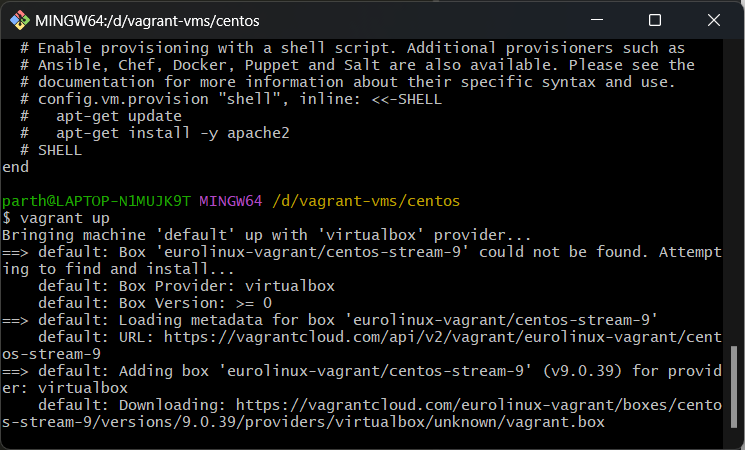
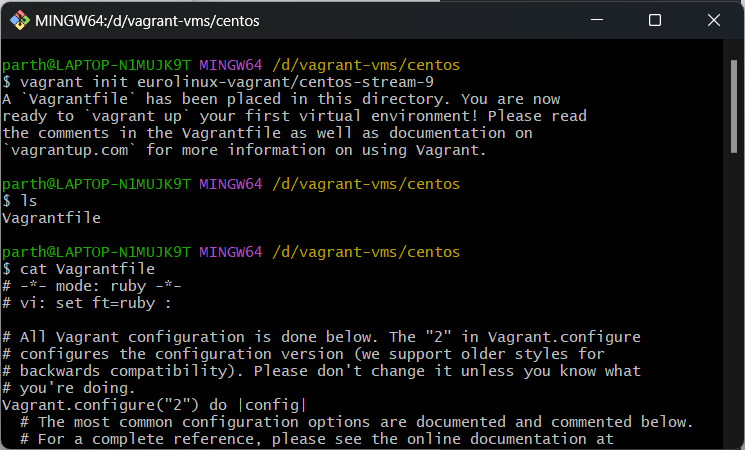
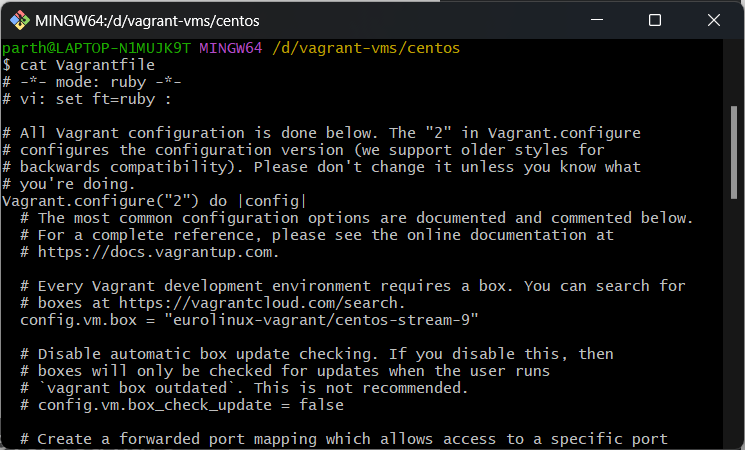
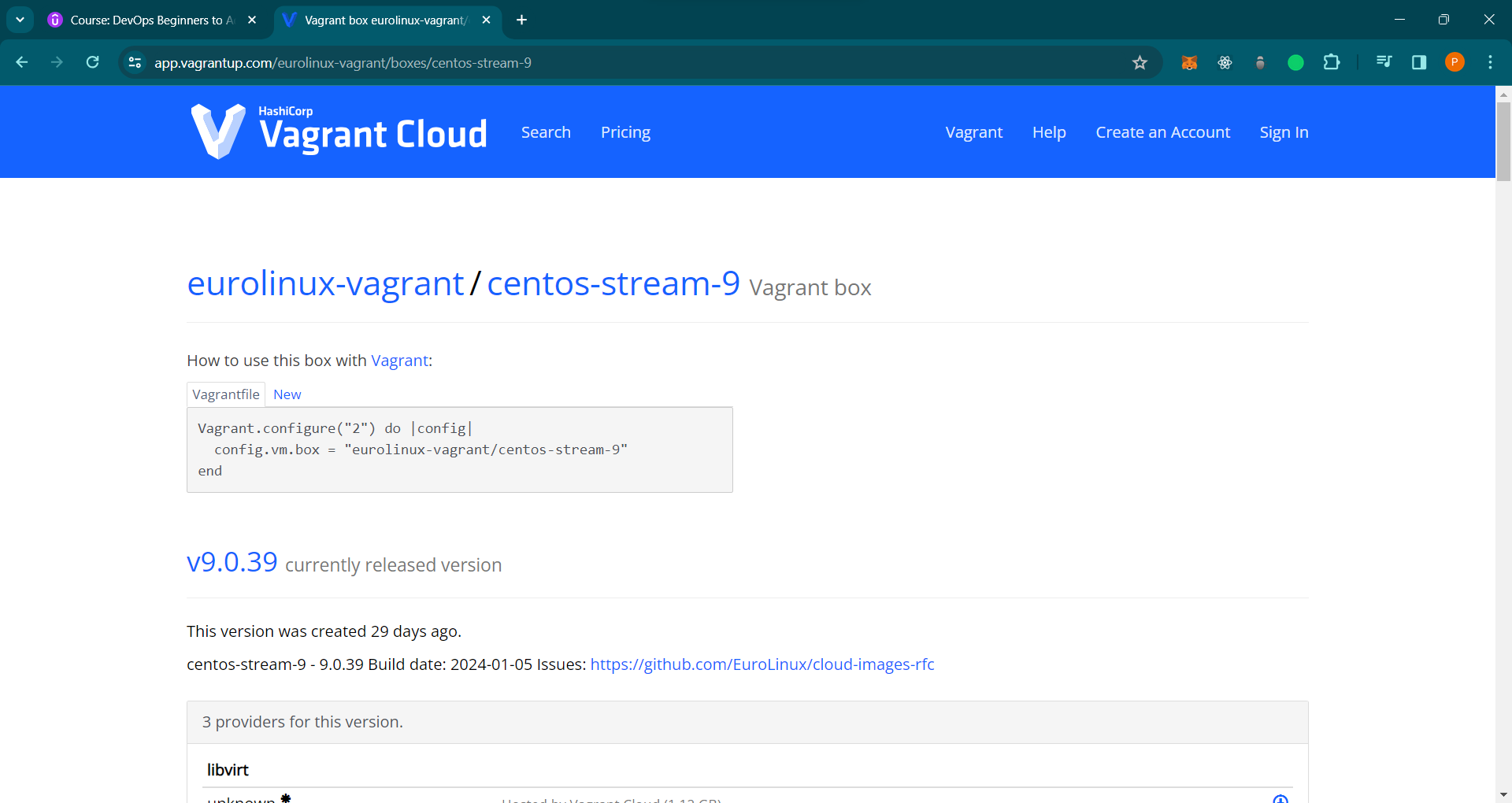
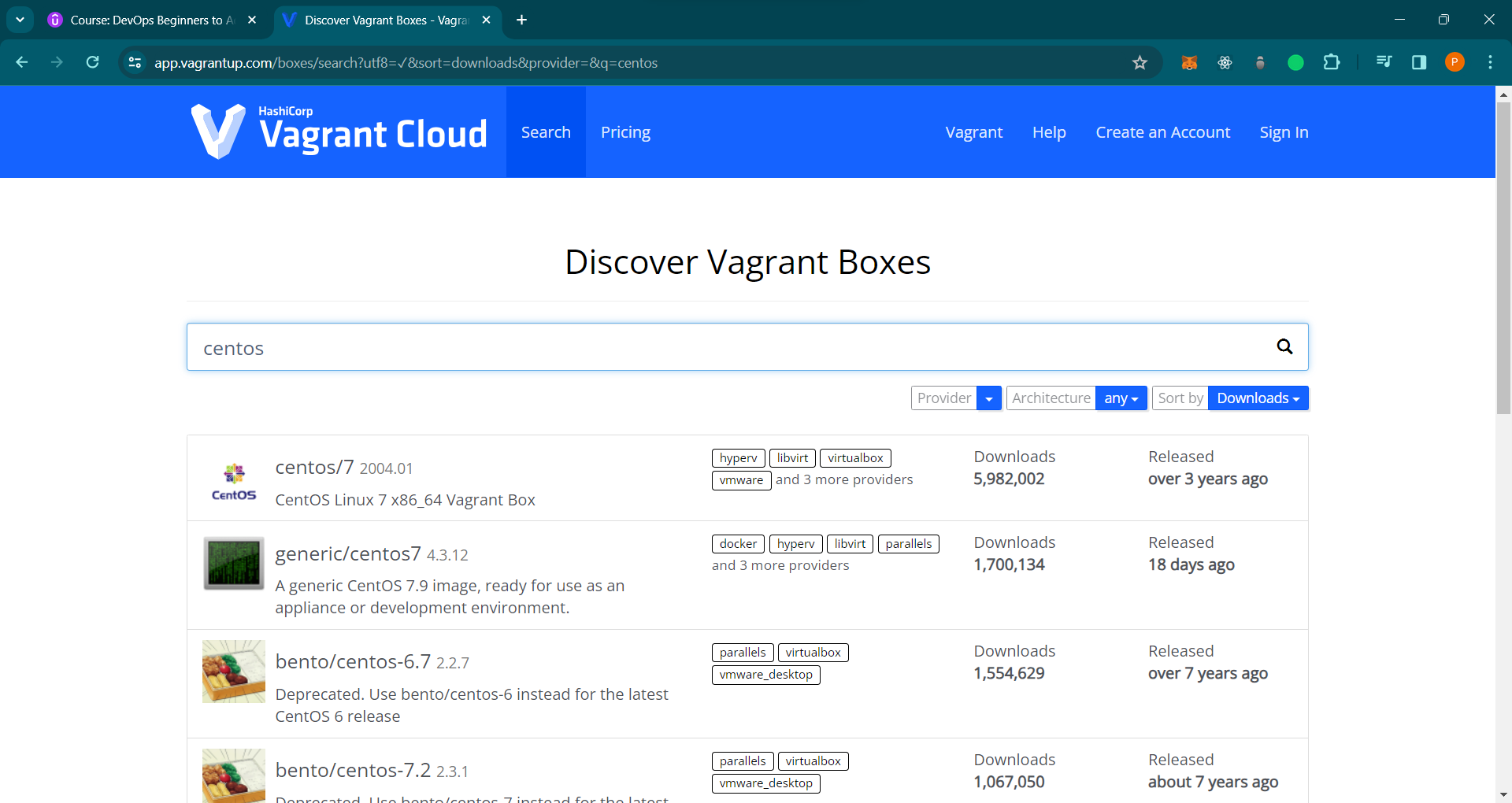
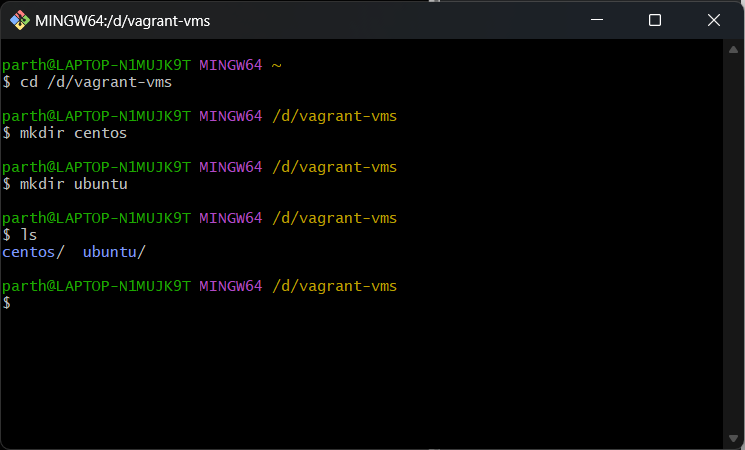
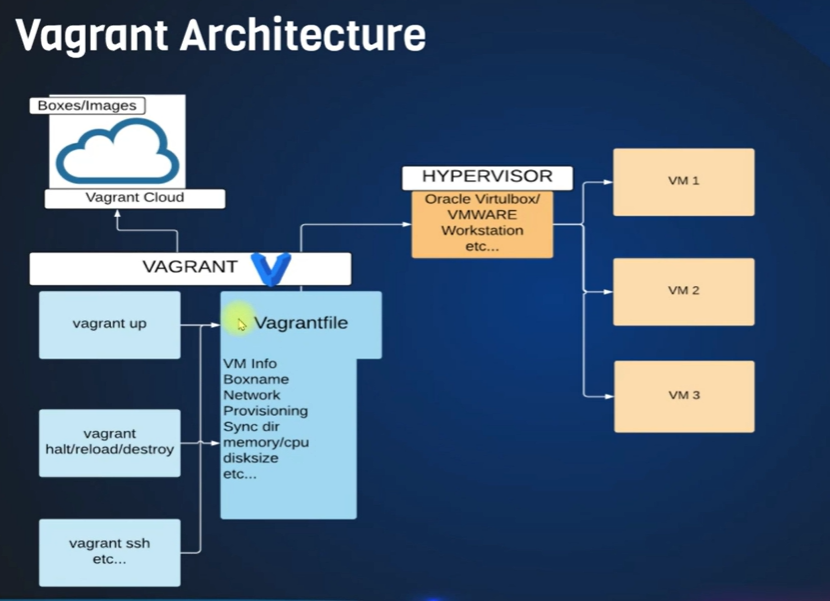
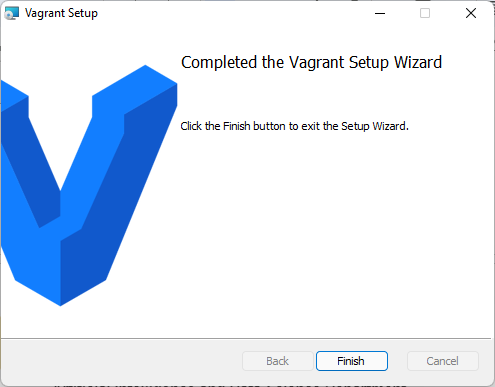
Vagrant provides easy to configure, reproducible, and portable work environments built on top of industry-standard technology and controlled by a single consistent workflow to help maximize the productivity and flexibility of you and your team.

To achieve its magic, Vagrant stands on the shoulders of giants. Machines are provisioned on top of VirtualBox, VMware, AWS, or any other provider. Then, industry-standard provisioning tools such as shell scripts, Chef, or Puppet can automatically install and configure software on the virtual machine.

****

****

****

****

**Conclusion :** In conclusion, the exploration of DevOps principles, practices, and the role of a DevOps engineer, coupled with hands-on experience in installing Ubuntu and CentOS Linux using Vagrant, provides valuable insights into the collaborative, automated, and continuously improving world of DevOps. This knowledge and practical exposure equip individuals with essential skills for navigating the dynamic landscape of modern software development and IT operations.