Date 13 June 2023
Antonine Guerrier
Kumar Batch May21st

Statements

AppleBite & Co. is using Cloud for one of their products. The project uses modular components, multiple frameworks and want the components to be developed by different teams or by 3rd party vendors.
The company's goal is to deliver the product updates frequently to production with High quality & Reliability. They also want to accelerate software delivery speed, quality and reduce feedback time between developers and testers.
As development progressed, they are facing multiple problems, because of various technologies involved in the project. Following are the problems:

• Building Complex builds is difficult
• Incremental builds are difficult to manage and deploy.

To solve these problems, they need to implement Continuous Integration & Continuous Deployment with DevOps using following tools:

Git
For version control for tracking changes in the code files

Jenkins
For continuous integration and continuous deployment

Docker
For deploying containerized applications

Ansible
Configuration management tools

This project will be about how to do deploy code to dev/stage/prod etc, just on a click of button. Link for the sample PHP application:

https://github.com/edureka-devops/projCert.git

Business challenges

As soon as the developer pushes the updated code on the GIT master branch, a new test server should be provisioned with all the required software. Post this, the code should be containerized and deployed on the test server. The deployment should then be built and pushed to the prod server. All, this should happen automatically and should be triggered from a push to the GitHub master branch.

Approach to solve

We need to develop a CI/CD pipeline to automate the software development, testing,

Packaging, and deployment reducing the time to market of the app and ensuring good quality service is experienced by end users. In this project, we need to

- Push the code to our GitHub repository.
- Create a continuous integration pipeline using Jenkins to compile, test, and package the code present in GitHub.
- Write Dockerfile to push the war file to the Tomcat server
- Integrate Docker with Ansible and write the playbook.

Below are the explanation on how these tools work.

Git

Developers need to update the code push it on the Git Master branch then a new test server will be provisioned with the required software. Post the code to be containerized and deployed it on the test server. The deployment should then be built and pushed to the prod server.

All this process should happen automatically and should be triggered from a push to the GitHub master branch. Before making any update, they would have to check that they have the latest up-to-date code by pulling the latest changes from remote repository. Then, they would create a new branch to isolate their changes, submit them for review before merging them into the master branch.

They would modify, or add the necessary files in the local repository, then add the modified files to the staging area using these commands:

- $ git pull origin master
- $ git checkout -b my-branch-name
- $ git add

Noticed that the .git means all changes in the current directory should be added to add specific files they should replace the . with the file paths. Then, commit the changes, create the commit with a descriptive message.

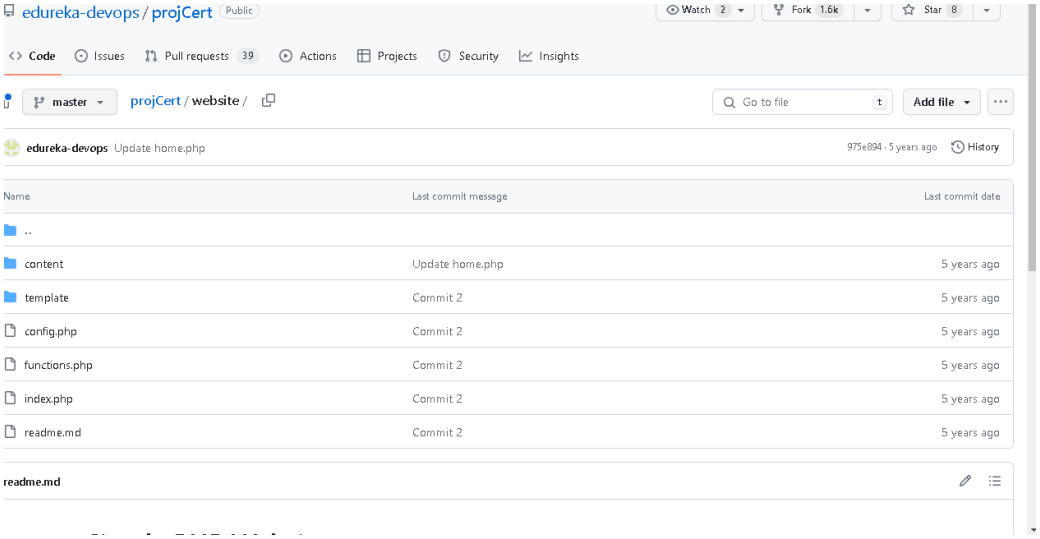
$ git commit -m “commit message”

Finally push the changes from the local branch to the remote repository:

$ git push origin and the branch name

Solutions:

They would connect to GitHub link to get the DevopsCode-Demo



Jenkins

Developers would install Jenkins on their preferred server, access the web interface to complete the initial setup. They would navigate through their projects to configure source code management system which include providing the git repository URL and credentials.

Create a new Jenkins pipeline job to build and test the code. Configure the build job to execute the necessary build commands or scripts, such as compiling code, running tests, or generating artifacts! Jenkins can be integrated with Docker, AWS, Azure, K8s, it is a CI tools can almost be integrated with everything.

Using plugins Jenkins will build and test:

- Manage plugins

- manage available plugins – install plugins lie maven which needs to test review and build the artefact. Maven package would be installed to facilitate the CI/CD) by providing a build automation and dependency management tool for Java projects.

Build now - Go to console output.

When create a new project - click on new item - enter item name e.g., using a declarative pipeline, test review and package the artefact. Developers are facing multiple problem because of building Complex difficult build and incremental builds that are difficult to manage and deploy. They would need to use a distributive pipeline.

What is a distributed pipeline would do?

In the context of CI/CD it would offer a setup where the various stages of the pipeline are distributed across multiple machines or agents. This would help to process and scale the CI/CD process. As a result, improves efficiency and reduce build and deployment times. As an example of a set up:

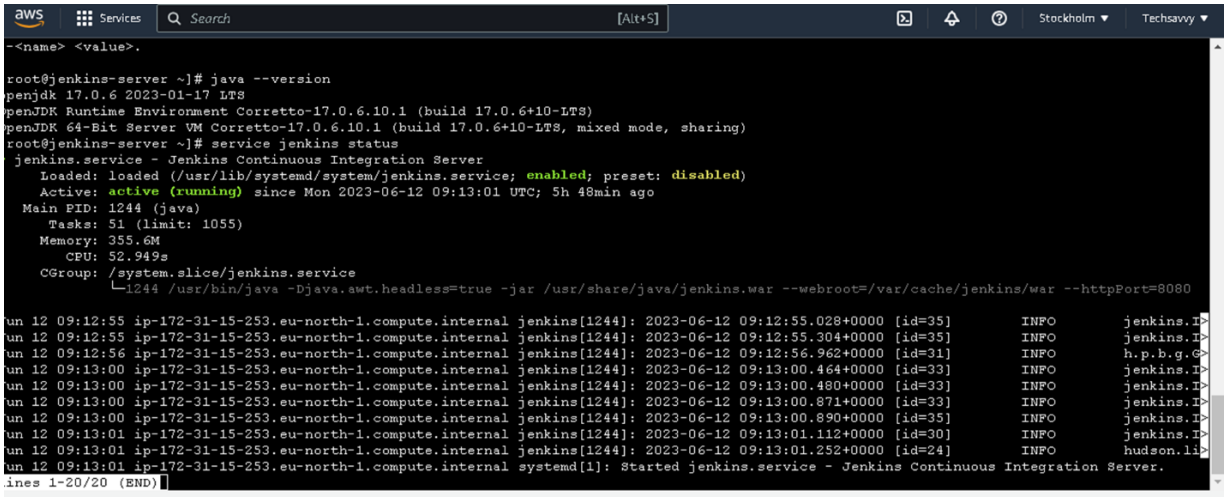
- Multiple build agents or separate machines or virtual environments would execute the build process. With each have tools and dependencies installed to perform the build tasks.
- Configuring the CI/CD system, such as Jenkins, to distribute the build workload across the multiple build agents. Configuring build jobs to run on specific agents or allowing the system to automatically distribute the workload based on what is available and workload balancing.
- The parallel testing in your CI/CD pipeline would enable the testing stages to distribute the testing workload across multiple agents. This can also be split or assign to different agents.
- Central artefact storage system such as cloud-based repository, where build artefact can be stored and accessed.
- Deployment agents responsible for deploying the application to various environments, such as:
    a. Development
    b. Staging
    c. Production

Each agent should have the necessary configurations and access permissions to perform the deployments. Configure your CI/CD system to deploy to multiple environments allowing for simultaneous deployment to different environments, reducing the overall deployment time.

In the case of AppleBite resolving the issues using LoadBalancing could be resourceful. Monitoring and scaling would help too. Also, by implementing load balancing, complex builds can handle increased traffic, improve response times, prevent bottlenecks, and provide scalability and fault tolerance by adding or removing resources as needed.

In Ec2 instance – install java 11 jdk and setup Jenkins as a root-user:

- $ sudo su
- $ start jenkins on instance



Docker

Docker is an open source which allows to automate deployment, scaling, and management of application as a result cost efficient and provides workload balanced quality based. It offers a way to package software and its dependencies into standardized units called containers. Each container runs as an isolated and lightweight virtualized environment, which makes it easy to deploy and run applications consistently across different environments, such as development, testing, and production. These are other solution for AppleBite development issues to consider.

Containers are lightweight, portable, can encapsulate application and its dependencies. However, because of various issues that may occur e.g different environment such one is using windows the other is using maybe mac the code may not respond the same therefore in this context encapsulating means packing the code + bin + lib + any other necessary dependency before submitting to the support team. In return, this would make the application independent of the environment so that it can work anywhere.

There are many benefits of using containers: great performance, code platform independent, instant boot. It is an OS level Virtualisation technology which help with sharing resources/infrastructure without affecting each other's work. For example, AWS, Namespaces(Ns) and cgroups(CG): are technologies use by docker to make this happen, they are features of Linux. Namespaces provide a layer of abstraction. Although there are different type of Namespace: PID; MINT; IPC; NET; UTS; and USER.

Virtual machine vs Container, the benefit of using containers on top of Virtual machine is VM has its own operating system; Docker container do not need operating system. Docker container is lightweight, VM has Guest OS, Docker container does not, which means data can be easily move from 1 location to another. Container can run unlimited number of applications depending on the machine. Whereas Virtual machine can run a limited number of applications depending on the VM configuration.

The containers come with something calls container engine which will facilitate the communication between Docker container and the OS, is productive, can be configured easily, has tool like DockerSwam and service discoveries to tell you which machine is running and where, service security control the access of the containers to users.

Docker Architecture has Docker registry on the internet e.g "hub.docker.com" to get all the images E.g, nginx and containers. You have docker Client or CLI; Docker host or Docker Daemon, the engine where everything is running and Docker registry where you push all your content. It can be private or public.

Docker has multiple commands that covers a wide range of problems:

$ service docker status - to show is docker is active – is docker daemon has initialised

$ service docker start

$ docker images – showing you the images you have stored

$ docker run -d --name devtest -v myvol2:/app ngnix:latest

$ docker ps -a

$ docker exec -it cont/id /bin/bash

To delete images and containers:

$ docker rm -f + container-ID

$ docker rmi + image-Id

More commands:

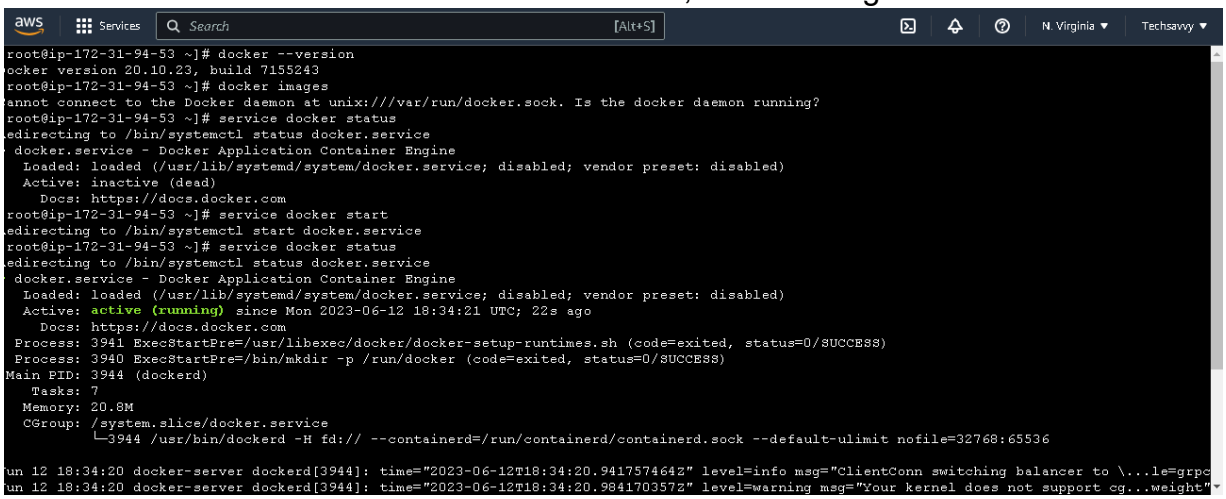
docker-compose up

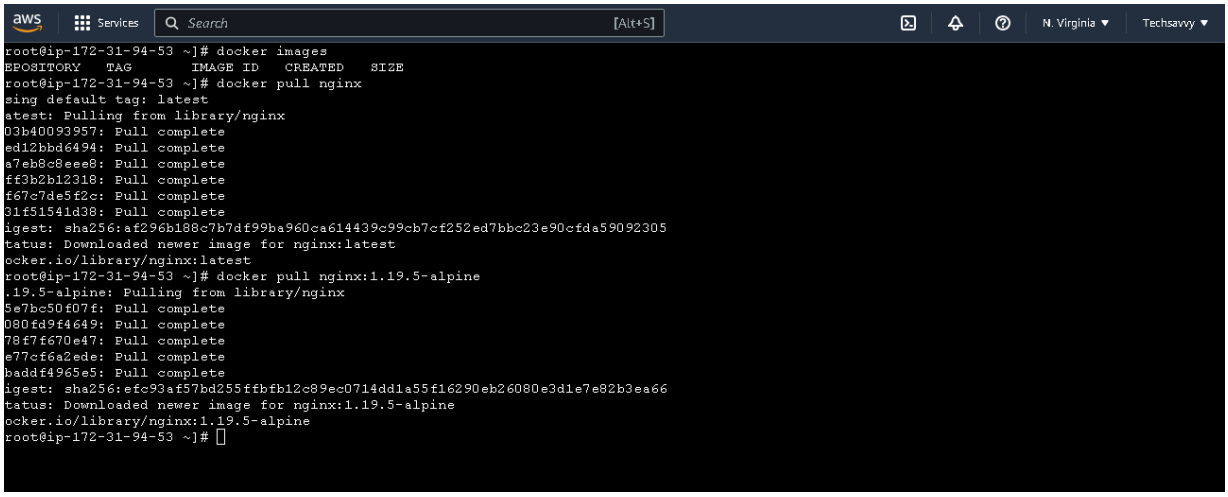
$ docker-compose down

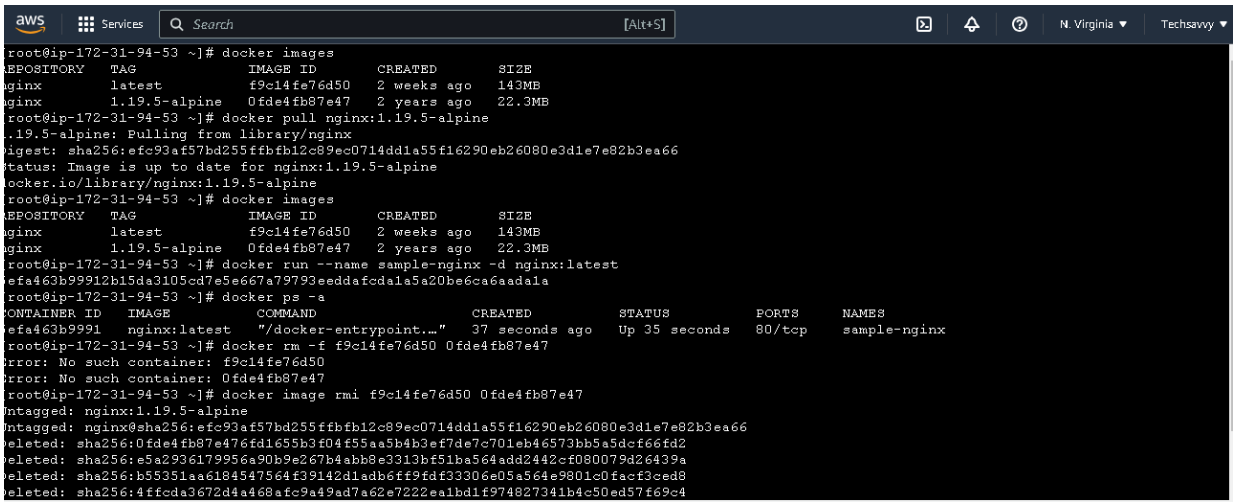
$ docker-compose up -d

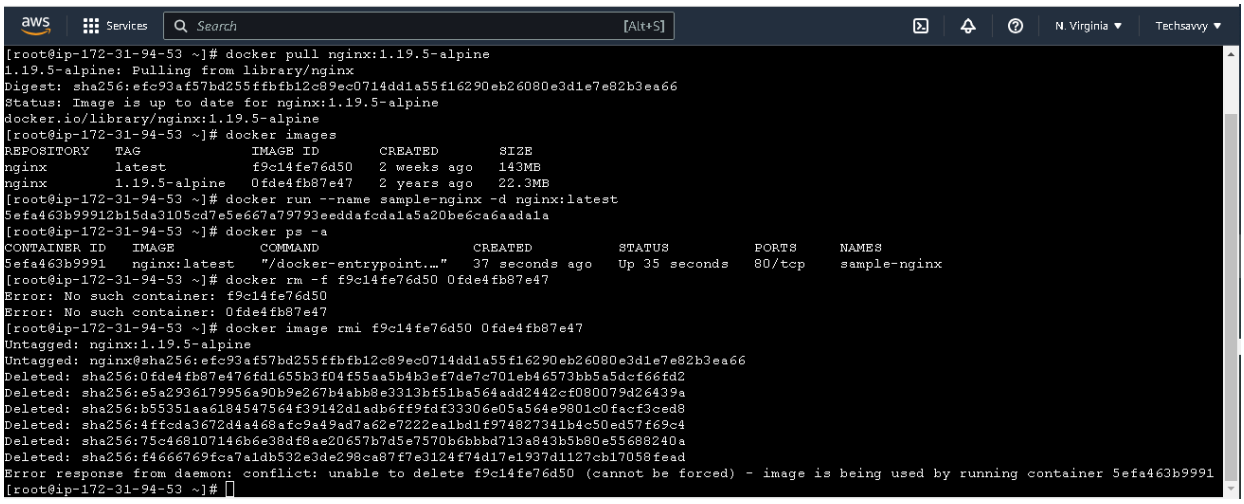
$ docker –help - display a wide range of commands and description

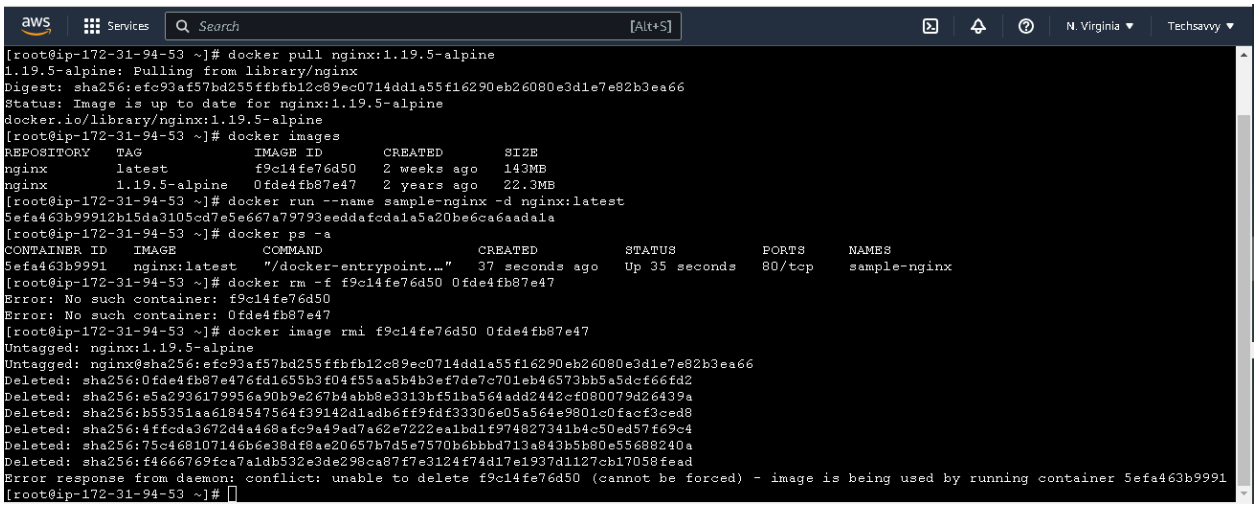
Ec2 – shows basic access to Docker installation; Docker images




aws
Services
Search
[Alt+S]
N. Virginia
Techsavvy
root@ip-172-31-94-53 ~]# docker images
EPOSITORY TAG IMAGE ID CREATED SIZE
root@ip-172-31-94-53 ~]# docker pull nginx
sing default tag: latest
atest: Pulling from library/nginx
03b40093957: Pull complete
ed12bbd6494: Pull complete
a7eb8c8eee8: Pull complete
ff3b2b12318: Pull complete
f67c7de5f2c: Pull complete
31f51541d38: Pull complete
igest: sha256:af296b188c7b7df99ba960ca614439c99cb7cf252ed7bbc23e90cfda59092305
tatus: Downloaded newer image for nginx:latest
ocker.io/library/nginx:latest
root@ip-172-31-94-53 ~]# docker pull nginx:1.19.5-alpine
.19.5-alpine: Pulling from library/nginx
5e7bc50f07f: Pull complete
080fd9f4649: Pull complete
78f7f670e47: Pull complete
e77cf6a2ede: Pull complete
baddf4965e5: Pull complete
igest: sha256:efc93af57bd255ffbfb12c89ec0714dd1a55f16290eb26080e3d1e7e82b3ea66
tatus: Downloaded newer image for nginx:1.19.5-alpine
ocker.io/library/nginx:1.19.5-alpine
root@ip-172-31-94-53 ~]#


aws
Services
Search
[Alt+S]
N. Virginia
Techsavvy
[root@ip-172-31-94-53 ~]# docker images
EPOSITORY TAG IMAGE ID CREATED SIZE
ginx latest f9c14fe76d50 2 weeks ago 143MB
ginx 1.19.5-alpine 0fde4fb87e47 2 years ago 22.3MB
[root@ip-172-31-94-53 ~]# docker pull nginx:1.19.5-alpine
.19.5-alpine: Pulling from library/nginx
igest: sha256:efc93af57bd255ffbfb12c89ec0714dd1a55f16290eb26080e3d1e7e82b3ea66
tatus: Image is up to date for nginx:1.19.5-alpine
ocker.io/library/nginx:1.19.5-alpine
[root@ip-172-31-94-53 ~]# docker images
EPOSITORY TAG IMAGE ID CREATED SIZE
ginx latest f9c14fe76d50 2 weeks ago 143MB
ginx 1.19.5-alpine 0fde4fb87e47 2 years ago 22.3MB
[root@ip-172-31-94-53 ~]# docker run --name sample-nginx -d nginx:latest
efa463b99912b15da3105cd7e5e667a79793eeddafcda1a5a20be6ca6aada1a
[root@ip-172-31-94-53 ~]# docker ps -a
ONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
efa463b9991 nginx:latest "/docker-entrypoint.…" 37 seconds ago Up 35 seconds 80/tcp sample-nginx
[root@ip-172-31-94-53 ~]# docker rm -f f9c14fe76d50 0fde4fb87e47
rror: No such container: f9c14fe76d50
rror: No such container: 0fde4fb87e47
[root@ip-172-31-94-53 ~]# docker image rmi f9c14fe76d50 0fde4fb87e47
ntagged: nginx:1.19.5-alpine
ntagged: nginx@sha256:efc93af57bd255ffbfb12c89ec0714dd1a55f16290eb26080e3d1e7e82b3ea66
eleted: sha256:0fde4fb87e476fd1655b3f04f55aa5b4b3ef7de7c701eb46573bb5a5dcf66fd2
eleted: sha256:e5a2936179956a90b9e267b4abb8e3313bf51ba564add2442cf080079d26439a
eleted: sha256:b55351aa6184547564f39142d1adb6ff9fdf33306e05a564e9801c0facf3ced8
eleted: sha256:4ffcda3672d4a468afc9a49ad7a62e7222ea1bd1f974827341b4c50ed57f69c4


aws
Services
Search
[Alt+S]
N. Virginia
Techsavvy
[root@ip-172-31-94-53 ~]# docker pull nginx:1.19.5-alpine
1.19.5-alpine: Pulling from library/nginx
Digest: sha256:efc93af57bd255ffbfb12c89ec0714dd1a55f16290eb26080e3d1e7e82b3ea66
Status: Image is up to date for nginx:1.19.5-alpine
docker.io/library/nginx:1.19.5-alpine
[root@ip-172-31-94-53 ~]# docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
nginx latest f9c14fe76d50 2 weeks ago 143MB
nginx 1.19.5-alpine 0fde4fb87e47 2 years ago 22.3MB
[root@ip-172-31-94-53 ~]# docker run --name sample-nginx -d nginx:latest
5efa463b99912b15da3105cd7e5e667a79793eeddafcda1a5a20be6ca6aada1a
[root@ip-172-31-94-53 ~]# docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
5efa463b9991 nginx:latest "/docker-entrypoint.…" 37 seconds ago Up 35 seconds 80/tcp sample-nginx
[root@ip-172-31-94-53 ~]# docker rm -f f9c14fe76d50 0fde4fb87e47
Error: No such container: f9c14fe76d50
Error: No such container: 0fde4fb87e47
[root@ip-172-31-94-53 ~]# docker image rmi f9c14fe76d50 0fde4fb87e47
Untagged: nginx:1.19.5-alpine
Untagged: nginx@sha256:efc93af57bd255ffbfb12c89ec0714dd1a55f16290eb26080e3d1e7e82b3ea66
Deleted: sha256:0fde4fb87e476fd1655b3f04f55aa5b4b3ef7de7c701eb46573bb5a5dcf66fd2
Deleted: sha256:e5a2936179956a90b9e267b4abb8e3313bf51ba564add2442cf080079d26439a
Deleted: sha256:b55351aa6184547564f39142d1adb6ff9fdf33306e05a564e9801c0facf3ced8
Deleted: sha256:4ffcda3672d4a468afc9a49ad7a62e7222ea1bd1f974827341b4c50ed57f69c4
Deleted: sha256:75c468107146b6e38df8ae20657b7d5e7570b6bbbd713a843b5b80e55688240a
Deleted: sha256:f4666769fca7a1db532e3de298ca87f7e3124f74d17e1937d1127cb17058fead
Error response from daemon: conflict: unable to delete f9c14fe76d50 (cannot be forced) - image is being used by running container 5efa463b9991
[root@ip-172-31-94-53 ~]#



There is also advantage of combining Docker with Apache as it can provide several benefits in terms of flexibility, scalability, and ease of deployment. Some of the advantages are:

- Containerization, encapsulate Apache and its dependencies into a container, provides a lightweight and isolated environment.
- Easy deployment, deploying an Apache web server becomes more streamlined and reproducible.
- Version control, Docker images can be visualised, and stored in a registry. Which in turn gives control over Apache versions and configurations. can track changes, roll back to previous versions, and ensure consistency across different deployments.
- Scalability, Isolation and security, environment consistency, rapid deployment and rollback and portability.

Ansible

Using ansible as configuration management tools offer lots of benefits. you to automate the configuration and management of systems, applications, and infrastructure. Using ansible's declarative language and playbook-based approach, you can define the desired state of your systems and let Ansible handle the tasks of achieving that state. This help reduce manual intervention, human error, and repetitive tasks.

Ansible has a simple syntax-based language YAML files, which makes it easy to read and write playbooks. The playbooks describe the desired configuration and steps to achieve it. Ansible does not require any specialized agents or additional software to be installed on managed nodes, making it straightforward to get started and maintain.

Ansible is agentless in a way that it communicates with managed nodes over SSH (Secure Shell) or other remote APIs.

Ansible allows you to treat your infrastructure as code. With Ansible playbooks, you can define, and version control your infrastructure configuration, making it easier to maintain, review changes, and collaborate with others.

Ansible ensures idempotent execution, meaning that even if you apply the same playbook multiple times it will result in a consistent state. It checks the current state of the system against the desired state defined in the playbook and only performs necessary changes.

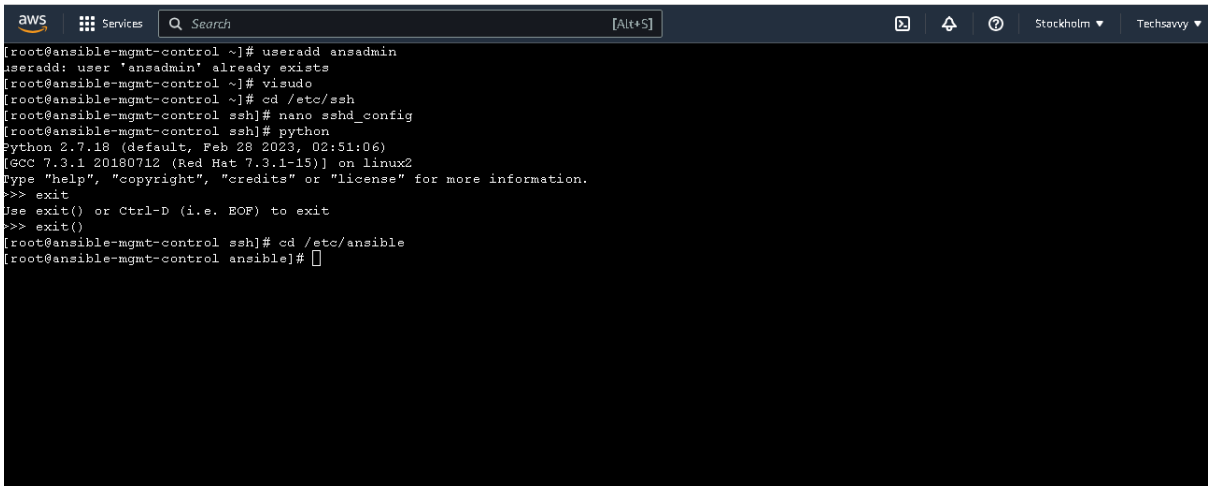
Ansible is designed to be scalable. As it can manage configurations across many systems simultaneously.

Ansible can support multiple operating systems making it easier to adapt to diverse infrastructure environments. and platforms such as Linux, Windows, macOS, or cloud platforms like AWS, Azure, or Google Cloud, providing modules and plugins to interact with various systems and services.

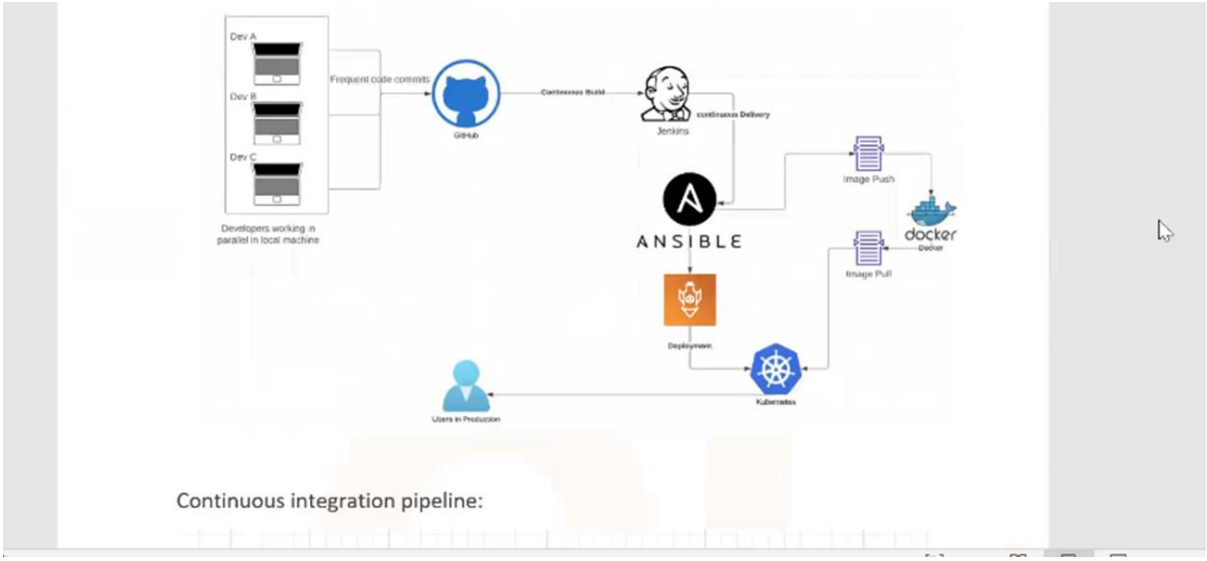
Ec2 – Installing ansible to give him power to allow him to configuration management and automation tasks on those instances. This has several advantages:

- Centralized management
- Automation at scale:

Ec2 Ansible basic commands



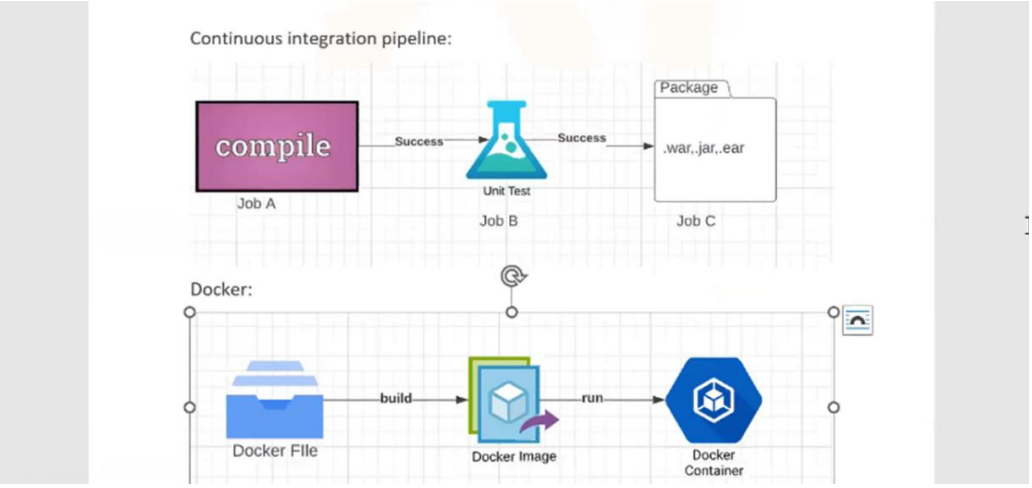
Ansible is OS automation tool that focuses on configuration management, application deployment, and task automation. It is data flow architecture which involves parsing the inventory, executing playbooks, performing tasks using modules, gathering facts, and providing detailed reporting. This flow allows Ansible to efficiently automate configuration management and deployment tasks while ensuring consistency and idempotent execution.



The PHP project project has been shared for usage. It is a maven project and has source and test folders created into it .It has a POM.xml file that list all the needed dependencies to execute this project.

Deliverables & Business benefits:

By incorporating Ansible into CI/CD processes, organizations can achieve faster and more reliable software releases, reduce manual intervention, and ensure consistent deployments across environments.



To summarise

Overall, together, Git, Jenkins, Docker, and Ansible provide a powerful set of tools for efficient software development, continuous integration, and deployment. They facilitate version control, automate build processes, provide containerization capabilities, and automate infrastructure provisioning and configuration

management. These tools contribute to improved collaboration, faster software releases, and increased reliability in the working environment.