Docker & Kubernetes

Daniel Mohanadhas
College of Science
The University of Texas at San Antonio
San Antonio. United States
daniel.mohanadhas@my.utsa.edu

Sukhjit Bal
College of Business
The University of Texas at San Antonio
San Antonio. United States
sukhjit.bal@my.utsa.edu

David J. Gulde
College of Science
The University of Texas at San Antonio
San Antonio. United States
David.Gulde@my.utsa.edu

Abstract- This paper details a comprehensive exploration of Docker, Kubernetes, and Meta's Llama 2. The team aimed to deploy a containerized web application utilizing these technologies on an Amazon EC2 cluster. Challenges encountered, such as the size of Llama 2 models, led to the strategic adoption of quantized models. The successful deployment on AWS underscores the team's adeptness with contemporary technologies, emphasizing adaptability and creative problem-solving. This project showcases the practical application of cloud computing, containerization, and artificial intelligence, providing valuable insights into modern software development.

Keywords- Docker, Llama 2, Amazon EC2, Containerization, Cloud Computing

I. Introduction

Group 3 consists of Daniel Mohanadhas, Sukhjit Bal, and David Gulde. With a shared passion for innovative technology, Group 3 decided to dedicate their final project to mastering Docker, Kubernetes, and Meta's artificial intelligence (AI) tool, Llama 2. The project's overall goal is to install and configure Kubernetes to manage docker containers on a cluster of VMs and install Llama 2 within the container. There are two main goals Group 3 wants to achieve in this project. First, is to learn more about Docker, Kubernetes, and Llama 2. The second goal is to develop a containerized web application to interact with Llama and deploy it on Kubernetes cluster. The required testbed and software tools used in this project are Amazon's EC2. Llama 2. Kubernetes v1.24, and Docker.io. This report will review the background, steps taken in the project, results, and conclusion.

II. Background

A. Docker

Docker is a powerful platform that simplifies the development, deployment, and scaling of applications through containerization. At its core, Docker utilizes container technology to encapsulate an application and its dependencies into a standardized unit called a container. These containers are isolated, lightweight, and portable, enabling developers to package their applications along with all required libraries and dependencies, ensuring consistency across different environments. Docker facilitates the creation of these containers through Docker images, which are defined by Dockerfiles—text files specifying the configuration and instructions for building an image. Once created, these Docker images can be shared and distributed through Docker Hub, a cloud-based registry. Docker's versatility extends to various operating systems and cloud platforms, allowing for seamless integration into diverse development and deployment workflows. Its impact is particularly notable in enabling efficient and scalable microservices architectures, where containers can be orchestrated and managed collectively to achieve an elevated level of agility and resource utilization. Overall, Docker has become an integral tool in modern software development, providing a standardized and efficient approach to building, shipping, and running applications.

B. Kubernetes

Kubernetes, also known as K8s, is an open-source container orchestration platform designed to automate the deployment, scaling, and management of containerized applications. The benefits of using Kubernetes can be reduced operational overhead, increased application uptime, improved scalability, and reduced costs. Popular use cases for Kubernetes include web applications, microservices, batch processing, and DevOps. Overall, Kubernetes is a powerful and versatile platform that can be used to manage a wide variety of containerized applications. It is a popular choice for businesses of all sizes, from startups to enterprises.

C. LLama 2

Llama 2 is a family of open-source large language models (LLMs) developed by Meta AI, the research division of Meta (formerly Facebook). It is a powerful tool for generating text, translating languages, writing various kinds of creative content, and answering questions in an informative way. Llama 2 is free for research and commercial use, making it an accessible option for developers and researchers around the world. Some key features of Llama 2 include large scalability, robust performance, versatility, and that it is open source. Some specific use cases for Llama 2 include chatbots like ChatGPT, translating languages, creative writing, and simply just the ability to answer questions.

D. Amazon EC2

Amazon Elastic Compute Cloud, more commonly known as EC2, web service that provides scalable, on-demand compute capacity in the cloud. It allows users to launch and manage virtual computers, or instances, on which they can run their applications. EC2 instances are available in a wide range of configurations, including different CPU, memory, and storage options. Users can also choose from a variety of operating systems, including Amazon Linux, Microsoft Windows, and Ubuntu. Some benefits of Amazon EC2 are scalability, cost-effectiveness, reliability, security, and flexibility. Some common use cases for Amazon EC2 are web hosting, application development, big data analytics, and high-performance computing. Amazon EC2 is a powerful and flexible service that can be used for a wide variety of applications. It is a popular choice for businesses of all sizes, from startups to enterprises.

III. Steps Taken

1. Create an application which uses llama2

The application idea of making a chatbot comes with llama2 being an LLM (large language models). This first required applying and getting access to the models themselves, which was a quick process. Facebook paired with hugging faces to allow easy access to download those models, as well as provided code examples for how to use them. These provided examples were then augmented to create a chat history of 2 queries for the application.

2. Containerize and make the application a webapp

The above application was then containerized and made into a webapp, but the webapp could only accept queries from either a POST or GET request and would send the output to local host. The main issue at this stage was the chat history of 2 queries, which I was not able to expand upon, and while not a deal-breaker, was far from ideal for a chatbot. While looking for how to expand the history, I found the state-of-the-art webui which is used for LLM's, which takes care of a chat history, and provides many other quality of life features like selecting parameters for models. With approval from the professor, I then used and containerized this webui going forward.

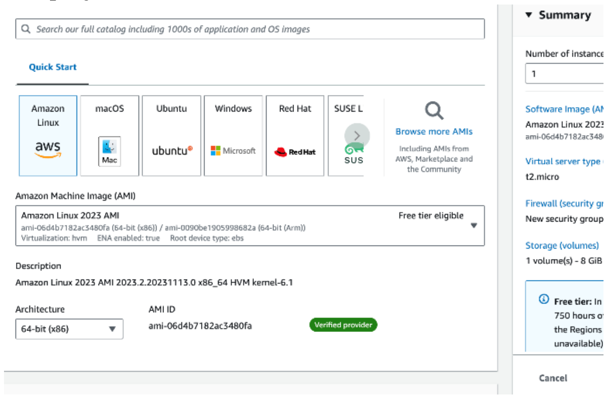
3. Deploy the Docker Image on the Cloud

We decided to choose AWS (Amazon Web Services) for where to deploy; it required 2 further steps: the selection and installation of the VM.

> *1. Get a Virtual Machine Image*
>
> As stated in the background, this project required the team to use Amazon's EC2 as the source of the project's Virtual Machine Instance. The Instance the team decided to use was "t2.micro." Amazon's Web Services provides "t2.micro" free for individuals for one year (750 hours total). Some of the stats of "t2.micro" include 1 vCPUs and 1.0 RAM (GiB). Figure 1 depicts what an individual would see on Amazon Web Services' website when trying to launch a "t2.micro".

T2 instances are a low-cost, general purpose instance type that provides a baseline level of CPU performance with the ability to burst above the baseline when needed. With On-Demand Instance prices starting at $0.0058 per hour, T2 instances are one of the lowest-cost Amazon EC2 instance options and are ideal for a variety of general-purpose applications like micro-services, low-latency interactive applications, small and medium databases, virtual desktops, development, build and stage environments, code repositories, and product prototypes [5]

The operating system used in this project was Amazon Linux as that was what the individuals in this project were most comfortable with.


Search our full catalog including 1000s of application and OS images
Quick Start
Amazon Linux
macOS
Ubuntu
Windows
Red Hat
SUSE L
Browse more AMIs
Including AMIs from AWS, Marketplace and the Community
Summary
Cancel

Amazon Machine Image (AMI)
Amazon Linux 2023 AMI   Free tier eligible
ami-06d4b7182ac3480fa (64-bit (x86)) / ami-0090be1905998682a (64-bit (Arm))
Virtualization: hvm   ENA enabled: true   Root device type: ebs

Description
Amazon Linux 2023 AMI 2023.2.20231113.0 x86_64 HVM kernel-6.1

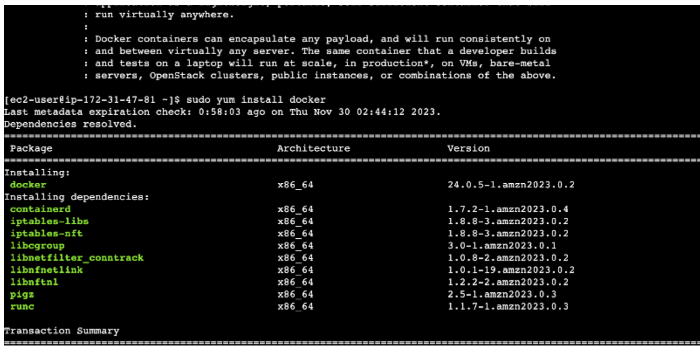
Architecture: 64-bit (x86)
AMI ID: ami-06d4b7182ac3480fa   Verified provider

Number of instance: 1
Software Image (AM: Amazon Linux 2023, ami-06d4b7182ac348
Virtual server type: t2.micro
Firewall (security gr: New security group
Storage (volumes): 1 volume(s) - 8 GiB
Free tier: In 750 hours o the Regions unavailable)

Figure 1.

2. Install Docker on to Virtual Machine

Once the "t2.micro" is set up, the next step is to install Docker on to it. Using the command "sudo yum install docker" will allow users on Amazon Linux to download the latest version of docker. Figure 2 depicts what a user would see post install of docker. Once installed, it is critical to make sure that Docker is on and running. To check the status of docker, run the "sudo service docker status" command. If Docker is not running, using "sudo service docker start" will turn on Docker, then the "sudo systemctl enable docker" command will ensure that Docker automatically start on reboots, and finally restarting the instance.

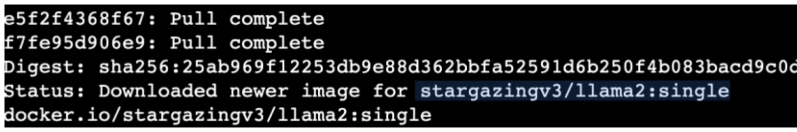


```
             : run virtually anywhere.
             :
             : Docker containers can encapsulate any payload, and will run consistently on
             : and between virtually any server. The same container that a developer builds
             : and tests on a laptop will run at scale, in production*, on VMs, bare-metal
             : servers, OpenStack clusters, public instances, or combinations of the above.

[ec2-user@ip-172-31-47-81 ~]$ sudo yum install docker
Last metadata expiration check: 0:58:03 ago on Thu Nov 30 02:44:12 2023.
Dependencies resolved.
==========================================================================================
 Package                                 Architecture          Version
==========================================================================================
Installing:
 docker                                  x86_64                24.0.5-1.amzn2023.0.2
Installing dependencies:
 containerd                              x86_64                1.7.2-1.amzn2023.0.4
 iptables-libs                           x86_64                1.8.8-3.amzn2023.0.2
 iptables-nft                            x86_64                1.8.8-3.amzn2023.0.2
 libcgroup                               x86_64                3.0-1.amzn2023.0.1
 libnetfilter_conntrack                  x86_64                1.0.8-2.amzn2023.0.2
 libnfnetlink                            x86_64                1.0.1-19.amzn2023.0.2
 libnftnl                                x86_64                1.2.2-2.amzn2023.0.2
 pigz                                    x86_64                2.5-1.amzn2023.0.3
 runc                                    x86_64                1.1.7-1.amzn2023.0.3

Transaction Summary
==========================================================================================
```

Figure 2.

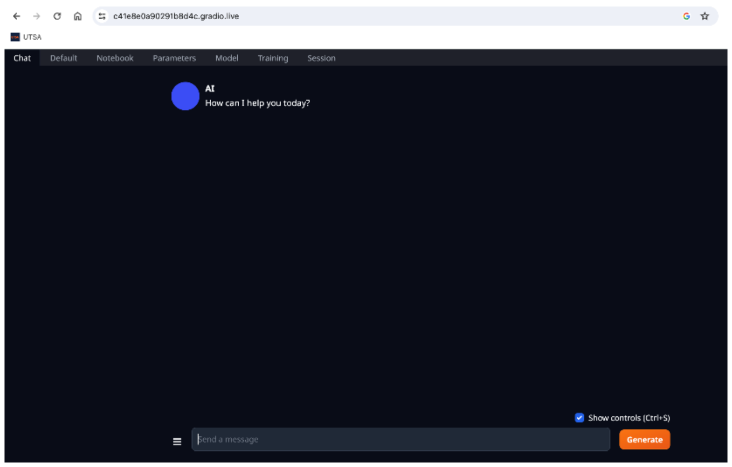
4. Pull Docker Image and Run it

The Final part of this project would be to pull the Docker Image that Daniel created and run it. Once confirmed that Docker is up and running on the VM, using the command "docker pull stargazingv3/llama2:single" will allow the VM to download the created image. Figure 3. shows what you should see once the image is downloaded successfully. The next step is to run the image. The command the team used was "docker run -it -e EXTRA_LAUNCH_ARGS="--share" -p 7860:7860 all stargazingv3/llama2:single". Once it finishes loading, the user will get a link that ends with ".gradio.live." The user will then need to grab that link and enter it in a web browser. The link should open a webpage where the user can interact with the container and ask questions as if it was a regular chat box. Figure 4 depicts how the chatbot will look once going to the provided URL.



```
e5f2f4368f67: Pull complete
f7fe95d906e9: Pull complete
Digest: sha256:25ab969f12253db9e88d362bbfa52591d6b250f4b083bacd9c0d
Status: Downloaded newer image for stargazingv3/llama2:single
docker.io/stargazingv3/llama2:single
```

Figure 3.



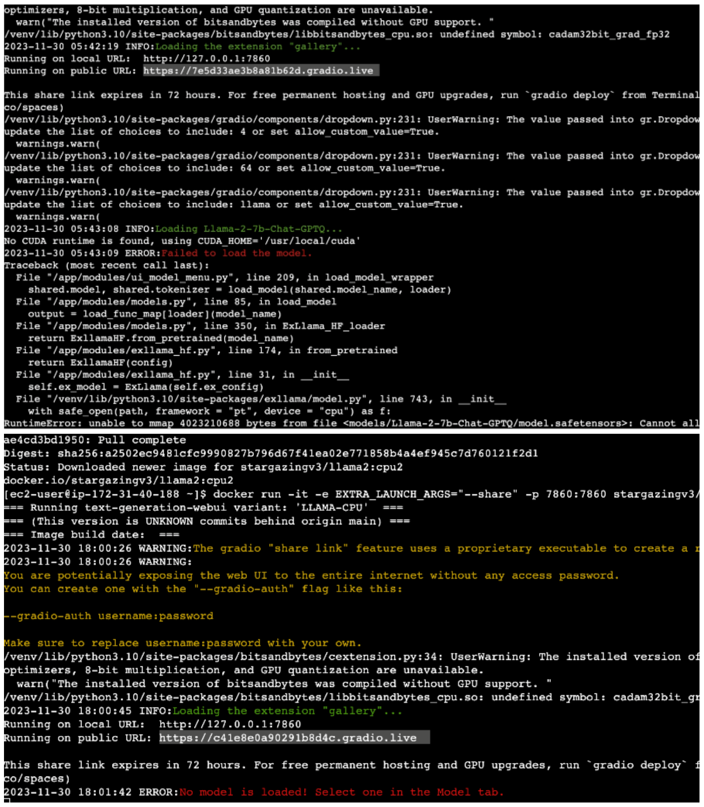
c41e8e0a90291b8d4c.gradio.live

AI
How can I help you today?

Figure 4.

IV. Outcomes

There were many issues which needed to be overcome right from step one: using the llama2 models. Facebook and Hugging Face may have made it easy to download and get started, the smallest model they offer is 13GB, which made things hard to run. The wsl environment would frequently crash when trying to build docker images. Figure 5 depicts what would happen when a user would try to interact with the Chatbot for both the Single and CPU instances. It shows that it is trying to load up Llama 2 when prompted a question, but then fails to load the model. The team was also unable to use Kubernetes due to CPU and memory constraints.
While searching for other llama2 models that could be used, the team came across quantized models. Quantized models are smaller than their regular counterparts, as they consist of less precise operations, like using FP16 instead of FP32. In addition, they can be modified to be CPU-only, called GGML models. These models can be significantly smaller, and the team found a 7 billion parameter one which was a little under 3 GB, which the team used.



```
optimizers, 8-bit multiplication, and GPU quantization are unavailable.
  warn("The installed version of bitsandbytes was compiled without GPU support. "
/venv/lib/python3.10/site-packages/bitsandbytes/libbitsandbytes_cpu.so: undefined symbol: cadam32bit_grad_fp32
2023-11-30 05:42:19 INFO:Loading the extension "gallery"...
Running on local URL:  http://127.0.0.1:7860
Running on public URL: https://7e5d33ae3b8a81b62d.gradio.live

This share link expires in 72 hours. For free permanent hosting and GPU upgrades, run `gradio deploy` from Terminal
co/spaces)
/venv/lib/python3.10/site-packages/gradio/components/dropdown.py:231: UserWarning: The value passed into gr.Dropdow
update the list of choices to include: 4 or set allow_custom_value=True.
  warnings.warn(
/venv/lib/python3.10/site-packages/gradio/components/dropdown.py:231: UserWarning: The value passed into gr.Dropdow
update the list of choices to include: 64 or set allow_custom_value=True.
  warnings.warn(
/venv/lib/python3.10/site-packages/gradio/components/dropdown.py:231: UserWarning: The value passed into gr.Dropdow
update the list of choices to include: llama or set allow_custom_value=True.
  warnings.warn(
2023-11-30 05:43:08 INFO:Loading Llama-2-7b-Chat-GPTQ...
No CUDA runtime is found, using CUDA_HOME='/usr/local/cuda'
2023-11-30 05:43:09 ERROR:Failed to load the model.
Traceback (most recent call last):
  File "/app/modules/ui_model_menu.py", line 209, in load_model_wrapper
    shared.model, shared.tokenizer = load_model(shared.model_name, loader)
  File "/app/modules/models.py", line 85, in load_model
    output = load_func_map[loader](model_name)
  File "/app/modules/models.py", line 350, in ExLlama_HF_loader
    return ExllamaHF.from_pretrained(model_name)
  File "/app/modules/exllama_hf.py", line 174, in from_pretrained
    return ExllamaHF(config)
  File "/app/modules/exllama_hf.py", line 31, in __init__
    self.ex_model = ExLlama(self.ex_config)
  File "/venv/lib/python3.10/site-packages/exllama/model.py", line 743, in __init__
    with safe_open(path, framework = "pt", device = "cpu") as f:
RuntimeError: unable to mmap 4023210688 bytes from file <models/Llama-2-7b-Chat-GPTQ/model.safetensors>: Cannot all
```

```
ae4cd3bd1950: Pull complete
Digest: sha256:a2502ec9481cfc9990827b796d67f41ea02e771858b4a4ef945c7d760121f2d1
Status: Downloaded newer image for stargazingv3/llama2:cpu2
docker.io/stargazingv3/llama2:cpu2
[ec2-user@ip-172-31-40-188 ~]$ docker run -it -e EXTRA_LAUNCH_ARGS="--share" -p 7860:7860 stargazingv3/
=== Running text-generation-webui variant: 'LLAMA-CPU'  ===
=== (This version is UNKNOWN commits behind origin main) ===
=== Image build date:  ===
2023-11-30 18:00:26 WARNING:The gradio "share link" feature uses a proprietary executable to create a r
2023-11-30 18:00:26 WARNING:
You are potentially exposing the web UI to the entire internet without any access password.
You can create one with the "--gradio-auth" flag like this:

--gradio-auth username:password

Make sure to replace username:password with your own.
/venv/lib/python3.10/site-packages/bitsandbytes/cextension.py:34: UserWarning: The installed version of
optimizers, 8-bit multiplication, and GPU quantization are unavailable.
  warn("The installed version of bitsandbytes was compiled without GPU support. "
/venv/lib/python3.10/site-packages/bitsandbytes/libbitsandbytes_cpu.so: undefined symbol: cadam32bit_gr
2023-11-30 18:00:45 INFO:Loading the extension "gallery"...
Running on local URL:  http://127.0.0.1:7860
Running on public URL: https://c41e8e0a90291b8d4c.gradio.live

This share link expires in 72 hours. For free permanent hosting and GPU upgrades, run `gradio deploy` f
co/spaces)
2023-11-30 18:01:42 ERROR:No model is loaded! Select one in the Model tab.
```

Figure 5.

V. Conclusion

This project focused on mastering Docker, Kubernetes, and Meta's AI tool, Llama 2. The project aimed to install and configure Kubernetes for managing Docker containers on an Amazon EC2 cluster, incorporating Llama 2 within the containerized web application. The team encountered challenges with the large size of Llama 2 models, leading to issues in the WSL environment and constraints preventing the use of Kubernetes. To overcome these hurdles, the team adopted quantized models, specifically a 7 billion parameter model, to reduce size and resource requirements. The successful deployment on AWS demonstrates the adaptability and problem-solving skills of the group in achieving their project goals. This endeavor showcases the practical application of cutting-edge technologies in a real-world context, emphasizing the importance of flexibility and innovation in addressing challenges in the rapidly evolving landscape of technology and artificial intelligence.

References

[1]
"Amazon Elastic Compute Cloud Documentation," Amazon.com, 2022. https://docs.aws.amazon.com/ec2/ (accessed Nov. 26, 2023).

[2]
Docker, "Docker Documentation," *Docker Documentation*, Oct. 31, 2019. https://docs.docker.com/

[3]
]"Kubernetes Documentation," Kubernetes.io, 2019. https://kubernetes.io/docs/home/

[4]
"Llama 2 - Resource Overview," Meta AI. https://ai.meta.com/resources/models-and-libraries/llama/

[5]
Amazon, "Amazon EC2 T2 Instances," https://aws.amazon.com/ec2/instance-types/t2/