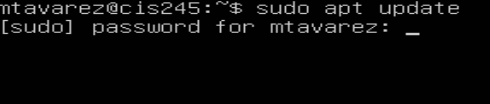
**Docker**

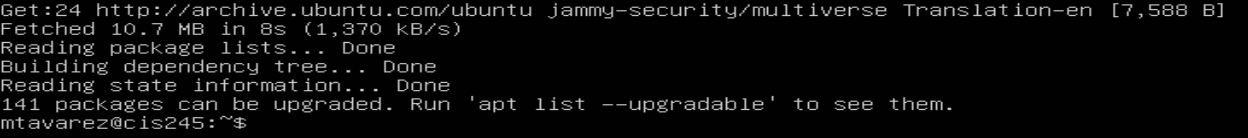
* Docker is a platform designed to make it easier to create, deploy, and run applications by using containers. Containers allow developers to package an application with all of its dependencies into a standardized unit for software development. With Docker, you can isolate your application from the environment it runs in, ensuring that it will run consistently regardless of where it's deployed. It's particularly popular in Linux environments due to its efficiency, lightweight nature, and flexibility.
* This is the process for installing docker in Ubuntu and CentOS:

**UBUNTU**

**Step 1: Update Package Index**

* Before installing Docker, ensure that your package index is up-to-date. This ensures you're fetching the latest versions of packages available. In order to accomplish this, run the command **sudo apt update:**

****

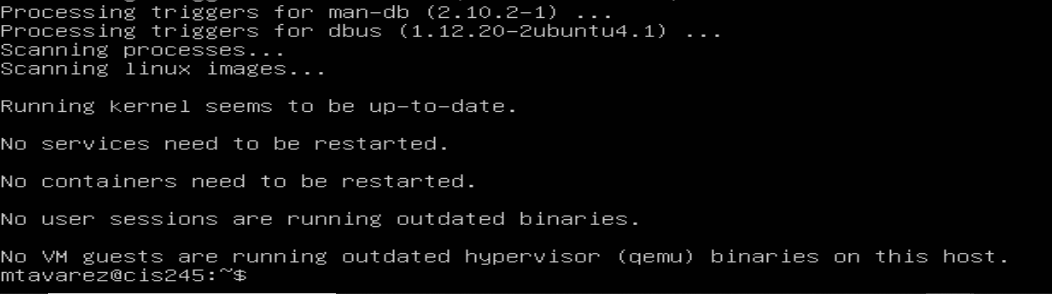


**Step 2: Install Required Packages**

* Docker requires some prerequisite packages to be installed. Install them using the following command:

**sudo apt install apt-transport-https ca-certificates curl software-properties-common**

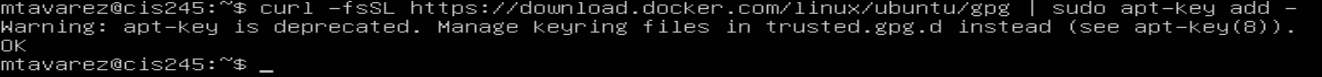
****

****

**Step 3: Add Docker’s Official GPG Key**

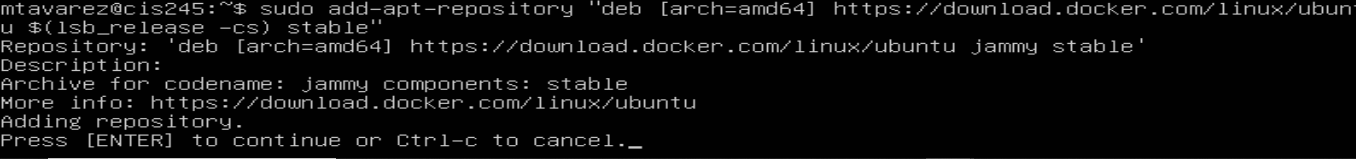
* To ensure the integrity of the Docker packages you download, you need to add Docker’s official GPG key. Use the next sequence for that:

**curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add –**

****

**Step 4: Add Docker Repository**

* Add the Docker repository to your system's sources list. This allows your package manager to find Docker packages. In order to get this working, follow the next command:

**sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable"**

**Step 5: Update Package Index Again**

* After adding the Docker repository, update your package index again to ensure it includes the Docker packages. This is the same step made in step 1.

**Step 6: Install Docker**

* Now you can install Docker using the following command:

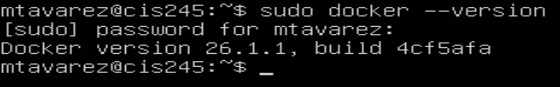
**sudo apt install docker-ce**

****

**Step 7: Verify Docker Installation**

* Check that Docker has been installed correctly by running:

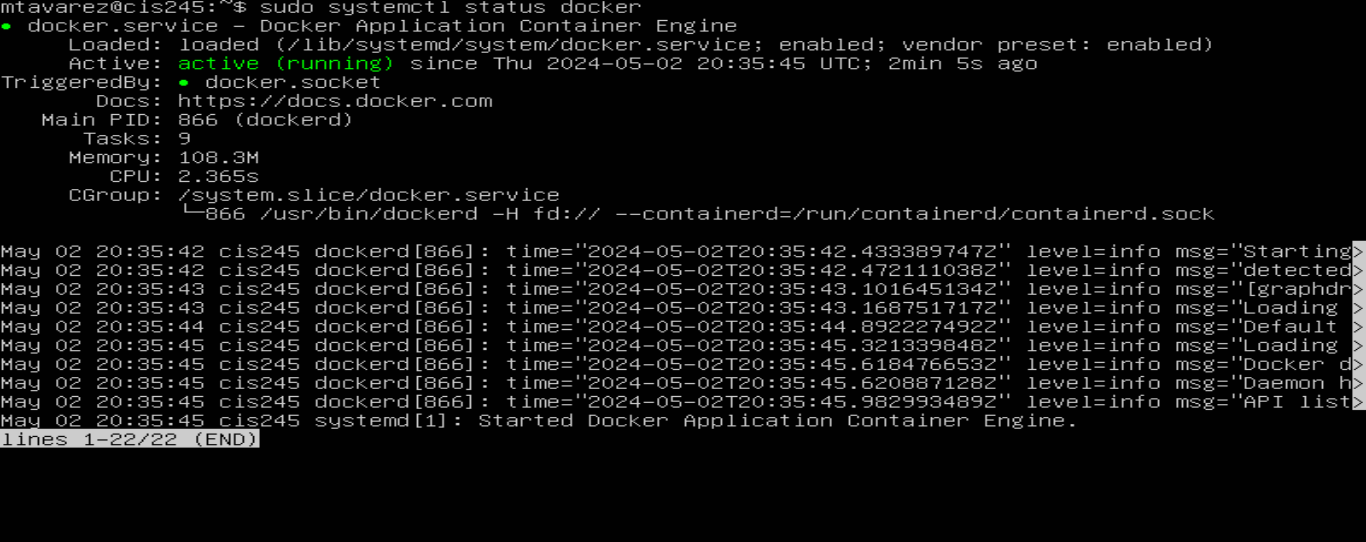
**sudo docker –version**

****

**Step 8: Start Docker Service**

* Docker should start automatically after installation, but you can ensure it's running by typing:

**sudo systemctl status docker**

****

If it's not running, you can start it with:

**sudo systemctl start docker**

**CentOS**

**Step 1: Update Package Index**

* Before installing Docker on CentOS, update your package index to ensure you're fetching the latest versions of packages available. In order to make this happen, use the next command:

**sudo yum update**

**Step 2: Install Required Packages**

* Install the necessary packages required for Docker installation on CentOS. Use the next command:

**sudo yum install -y yum-utils device-mapper-persistent-data lvm2**

****

**Step 3: Add Docker Repository**

* Add the Docker repository to your system's configuration. Follow the next command:

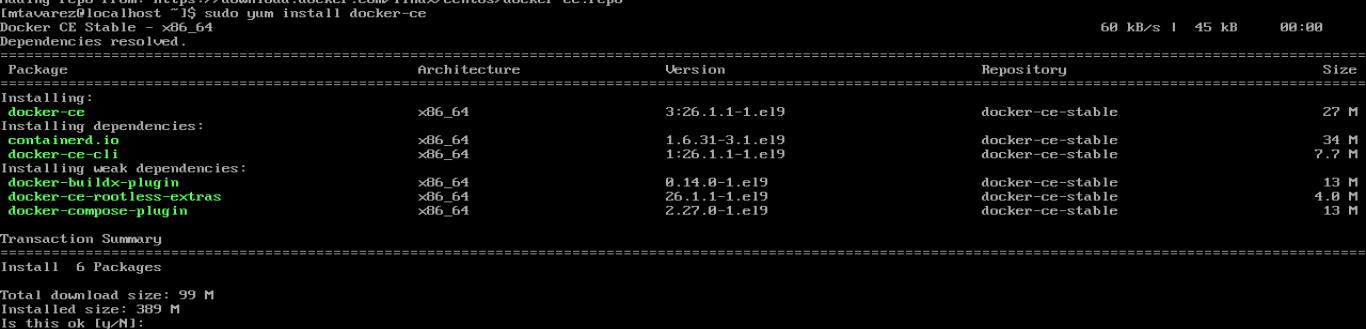
**sudo yum-config-manager --add-repo** [**https://download.docker.com/linux/centos/docker-ce.repo**](https://download.docker.com/linux/centos/docker-ce.repo)

****

**Step 4: Install Docker**

* Now, you can install Docker on CentOS using the following command:

**sudo yum install docker-ce**

****

**Step 5: Start Docker Service**

* After installation, start the Docker service using the following command:

**sudo systemctl start docker**

****

**Step 6: Enable Docker to Start on Boot**

* Ensure Docker starts automatically upon system boot. Execute the next command:

**sudo systemctl enable docker**

****

**Step 7: Verify Docker Installation**

* Check that Docker has been installed correctly by running:

**sudo docker –version**



* In conclusion, Docker should now be installed on your Linux distribution. Remember to use **sudo** before each Docker command if you're not logged in as the root user or don't have root privileges.

**Executive Summary: Understanding Containers and Their Implications for Acme Corp**

Containers are a form of virtualization technology that allow for the isolation and packaging of software applications and their dependencies. Unlike traditional virtual machines, containers share the host operating system's kernel, which makes them lightweight and efficient.

**Pros of Containers:**

1. **Portability:** Containers encapsulate an application and its dependencies, making it easy to move across different environments, from development to production.
2. **Scalability:** Containers can be quickly deployed and scaled up or down to meet changing demands, facilitating efficient resource utilization.
3. **Isolation:** Each container operates independently, ensuring that applications remain isolated from one another, thus reducing the risk of conflicts and security breaches.
4. **Consistency:** Containers ensure consistent runtime environments, reducing the likelihood of "it works on my machine" issues.
5. **Resource Efficiency:** Containers consume fewer resources compared to virtual machines, allowing for higher density and better utilization of hardware.

**Cons of Containers:**

1. **Security Concerns:** While containers provide isolation, vulnerabilities within container images or misconfigurations can still pose security risks if not properly managed.
2. **Complexity:** Managing containerized environments requires understanding of container orchestration tools, networking, and storage configurations, which can add complexity to deployment and maintenance.
3. **Persistence:** By default, containers are ephemeral, meaning that any data stored within them is lost when the container stops. This requires additional solutions for persistent storage.
4. **Learning Curve:** Adopting containers may require training and upskilling for teams unfamiliar with containerization concepts and tools.

**Recommendation for Acme Corp:**

Considering the advantages and challenges of containerization, Acme Corp should carefully evaluate its specific needs and infrastructure before deciding whether to adopt containers. If Acme Corp has a dynamic and rapidly evolving environment, with a need for scalability, portability, and efficient resource utilization, containers could offer significant benefits. However, it's crucial for Acme Corp to invest in proper security measures, training, and infrastructure to effectively leverage containers while mitigating associated risks. Overall, containers represent a powerful tool for modernizing application deployment and management, but their adoption should be approached strategically to maximize their benefits.

**Microk8s**

* MicroK8s is a lightweight Kubernetes distribution by Canonical for local development and small-scale production on Linux systems. It offers a quick setup of essential Kubernetes components for testing and development without the complexity of a full-scale cluster.
* To install MicroK8s in Ubuntu and CentOS, you can follow these steps. Each distro in this documentation has its own section:

**Ubuntu:**

**Step 1 Update Your System**

* This step ensures that your Ubuntu system is up-to-date with the latest package information. In order to execute an update, run this command:

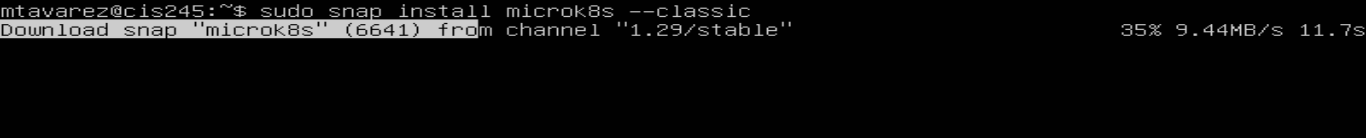
**sudo apt update**

****

**Step 2 Install MicroK8s**

* Installs MicroK8s using Snap package manager. Snap allows you to install packages and their dependencies securely and isolated from the rest of the system. Execute the next command:

**sudo snap install microk8s --classic**

****

****

**Step 3** **Check the Status**

* Verifies the status of the MicroK8s installation and waits until it's ready to use. Verify the status with the next command:

**microk8s status --wait-ready**

**A screenshot of a computer

Description automatically generated**

**Step 4** **Enable Required Add-ons**

* Activates essential add-ons like DNS, Dashboard, and Storage for MicroK8s. These add-ons provide additional functionality and tools for managing and deploying applications on the Kubernetes cluster. In this process, I decided to only enable dashboard, dns, and registry. Thus, to disable an add-on, it is the same process. Run the next command:

**microk8s enable “add-ons”**

**microk8s disable “add-ons”**

**A screenshot of a computer program

Description automatically generated**

**A black background with white text

Description automatically generated**

**A screenshot of a computer screen

Description automatically generated**

* To see other add-ons, execute this command:

**microk8s status**

* For help, execute:

**microk8s enable –help**

**Step 5** **Add Your User to the MicroK8s Group** (optional)

* Adds the current user to the microk8s group, allowing them to run MicroK8s commands without requiring sudo privileges. This step enhances convenience when working with MicroK8s. Use the next command to apply users to a group:

**sudo usermod -a -G microk8s $USER**

**CentOS**

**Step 1 Update Your System**

* Similar to Ubuntu, this step updates the CentOS system to ensure it has the latest package information. Execute the next command:

**Sudo yum update**

**Step 2 Enable Snapd**

* CentOS doesn't come with Snapd (Snap package manager) pre-installed. This step installs Snapd and enables its service to manage Snap packages. Execute the following commands in order to achieve this:

**sudo yum install snapd**

**sudo systemctl enable --now snapd.socket**

**A screenshot of a computer

Description automatically generated**

**A black screen with white text

Description automatically generated**

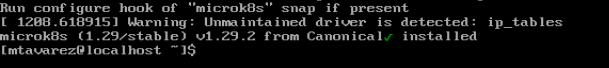
****

**Step 3** **Install MicroK8s**

* Installs MicroK8s using Snap package manager, just like in Ubuntu. Snap simplifies the installation process by handling dependencies and ensuring a consistent installation across different distributions. Execute the following:

**sudo snap install microk8s --classic**

****

****

**Step 4 Check the Status**

* Verifies the status of the MicroK8s installation and waits until it's ready to use, similar to the Ubuntu step:

**microk8s status --wait-ready**

**A screenshot of a computer

Description automatically generated**

**Step 5 Enable Required Add-ons**

* Activates essential add-ons (DNS, Dashboard, Storage) for MicroK8s, providing additional functionality for managing the Kubernetes cluster. Sames as the ubuntu, each individual add-on can be enable by preference of the user in charge. For this CentOS demonstration, I enabled DNS, Dashboard, and Registry:

**microk8s enable “add-ons”**

**microk8s disable “add-ons”**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

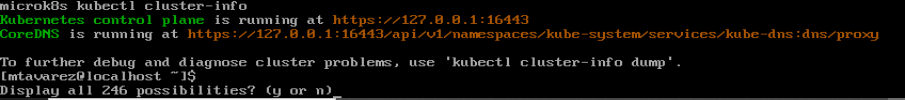
**A black background with white text

Description automatically generated**

Step 5 **Check the Cluster Info**

* Displays information about the Kubernetes cluster, such as API server endpoint, DNS service, and more. This helps ensure that the cluster is operational and accessible. Run the following command:

**microk8s.kubectl cluster-info**

****

**Step 6 Add Your User to the MicroK8s Group** (optional)

* Adds the current user to the **microk8s** group, allowing them to run MicroK8s commands without requiring sudo privileges. This step enhances convenience when working with MicroK8s. Just like Ubuntu, run the following command:

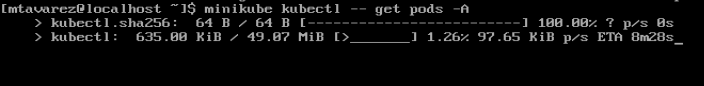
**sudo usermod -a -G microk8s $USER**

* In conclusion, These steps should guide you through the installation process of MicroK8s on both Ubuntu and CentOS systems.

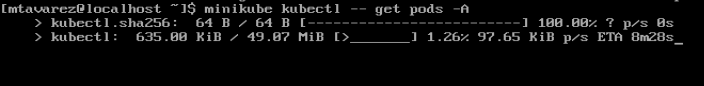
**Minikube tutorial**

* This was probably one of the hardest things to understand. Tried different approaches and resources, but in the end, it clicked just a little bit. This tutorial was very confusing, and at times I couldn’t understand the material because of the fact that I needed to go back and forward between different steps and areas that needed to be done before hand. One of these was in order to initialize Hello Minikube, I needed to go back to minikube start. Another instance where starting minikube, I needed to install other tools. It was very aggravating in a sense.
* In order to make some of the areas work, I first started with installing minikube based on my operating system. At first I couldn’t figure out what was I doing wrong, but eventually it worked. Once installed, I then proceeded to interact with the cluster by executing the next command:

**Minikube kubectl – get po -A**

****

* Once done, I then proceed to deploy some applications as a way to understand more about Kubernetes.



* Once the tools necessary where installed, I then proceed to start minikube and it did work and proceeded to deploy a test container to manages a pod:

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

* Furthermore, I added add-ons and learned how to disable them.

A screenshot of a computer

Description automatically generated

* In conclusion, the tutorial was a little confusing and messy at times, but it did what is supposed to do, show you and teach you how Kubernetes work.