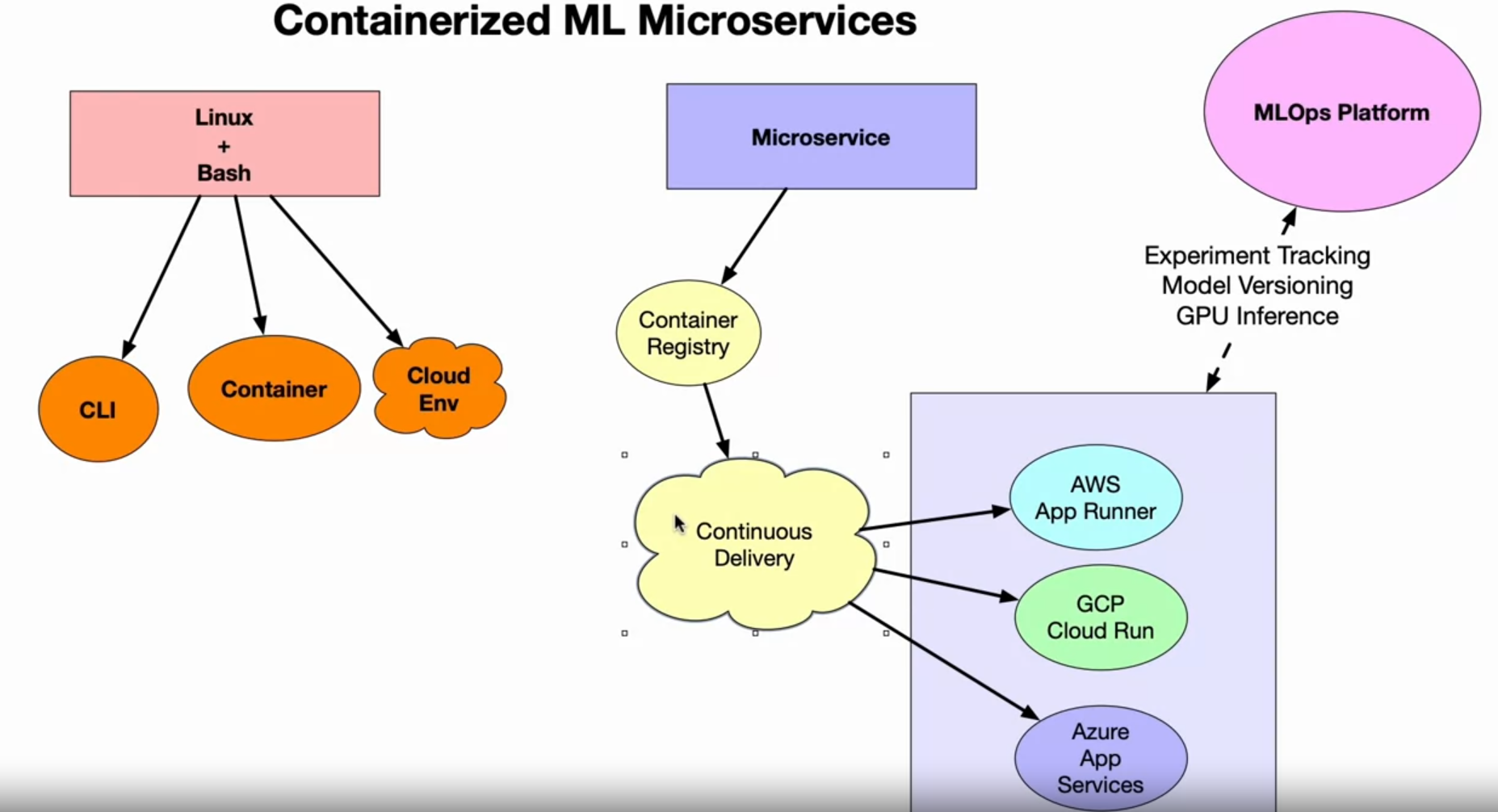
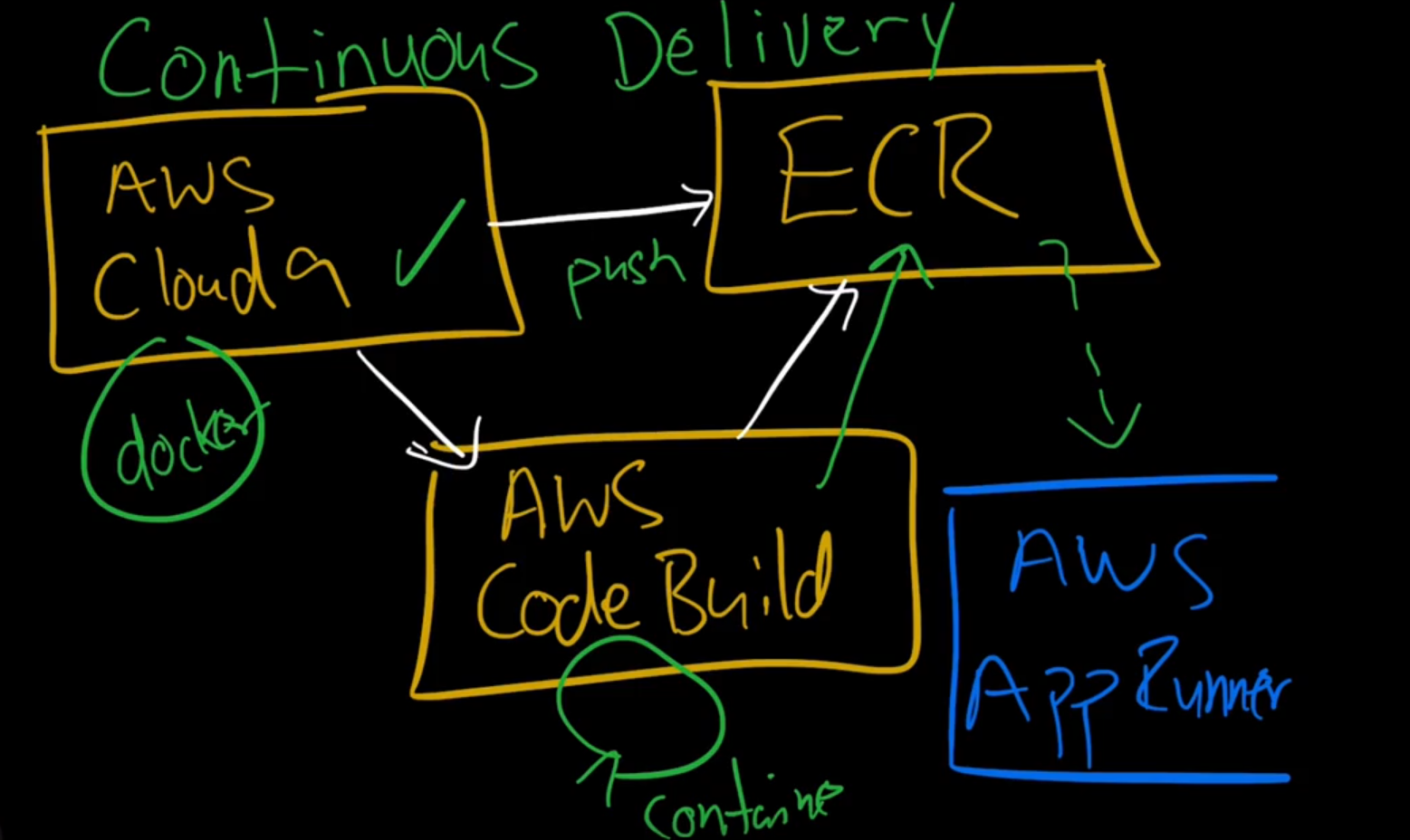
* **Container** - A standardized, lightweight software package that bundles together an application's code and dependencies to run reliably in any environment.
* **Container Registry** - A repository for storing, sharing and deploying container images, often integrated into CI/CD pipelines. Examples: Docker Hub, AWS ECR.
* **Container Orchestration** - Automated management, scaling and coordination of containers leveraging platforms like Kubernetes and Amazon ECS.
* **Continuous Delivery** - Software development practice to build, test and release containers quickly and reliably by automating deployments through CI/CD pipelines.
* **Infrastructure as Code** - Managing infrastructure like networking, compute, storage through machine-readable definition files rather than manual processes. Enables reproducibility.
* **Distroless Container** - Optimized container image containing only an app, run-time language, and essential system libraries, omitting shells, package managers, etc. Improves security.



Containerized Microservices = container registry + continuous delivery = app runner / cloud run / app service



**Method 1 : -**

Cloud 9 -> IDE for code dev using virtual env

Push code to ECR from Cloud 9 to create image of code

Create App runner from ECR image repo which performs continuous delivery

**Method 2** : -

Cloud 9 -> IDE for code dev using virtual env

Push code to code repo Code Commit / GitHub / GitLab

Connect code repo to Code Build

Containerization is a software deployment process that bundles an application’s code with all the files and libraries it needs to run on any infrastructure. Traditionally, to run any application on your computer, you had to install the version that matched your machine’s operating system. For example, you needed to install the Windows version of a software package on a Windows machine. However, with containerization, you can create a single software package, or [container](https://aws.amazon.com/containers/), that runs on all types of devices and operating systems.

* Docker

Docker, or Docker Engine, is a popular open-source container runtime that allows software developers to build, deploy, and test containerized applications on various platforms. Docker containers are self-contained packages of applications and related files that are created with the Docker framework.

* Linux

Linux is an open-source operating system with built-in container technology. Linux containers are self-contained environments that allow multiple Linux-based applications to run on a single host machine. Software developers use Linux containers to deploy applications that write or read large amounts of data. Linux containers do not copy the entire operating system to their virtualized environment. Instead, the containers consist of necessary functionalities allocated in the Linux namespace.

* Kubernetes

Kubernetes is a popular open-source container orchestrator that software developers use to deploy, scale, and manage a vast number of microservices. It has a declarative model that makes automating containers easier. The declarative model ensures that Kubernetes takes the appropriate action to fulfil the requirements based on the configuration files.

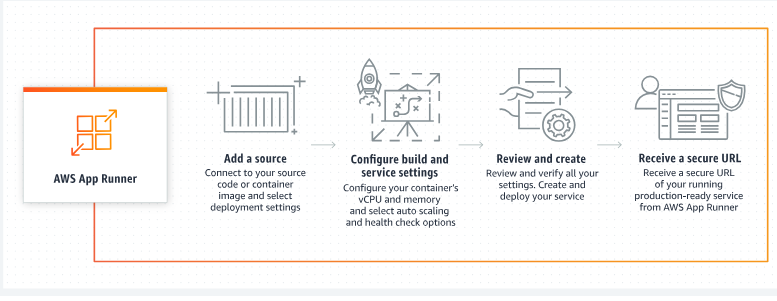
Containerization is a similar but improved concept of a VM. Instead of copying the hardware layer, containerization removes the operating system layer from the self-contained environment. This allows the application to run independently from the host operating system. Containerization prevents resource waste because applications are provided with the exact resources they need.

Serverless computing allows instant deployment of applications because there are no dependencies such as libraries or configuration files involved. The cloud vendor doesn't charge for computing resources when the serverless application is idle. Containers, on the other hand, are more portable, giving developers complete control of the application's environment.

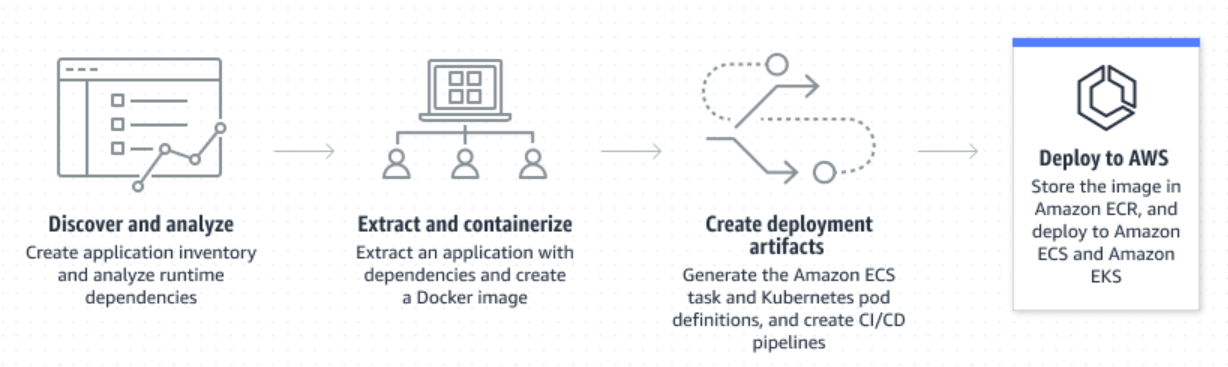
Cloud-native application development requires different technologies and approaches than conventional monolithic applications. Containerization is one of the technologies that allows developers to build cloud-native applications. It works with other cloud-native technologies, such as service mesh and APIs, to allow microservices to work cohesively in a cloud-native application.

***App Runner*** *: -*

* AWS App Runner is a fully managed service that makes it easy for developers to deploy from source code or container image directly to a scalable and secure web application
* App runner supports continuous delivery from container registry / code repository
* App runner is a serverless service which create ECS for containerization & Code Build for continuous delivery in backend
* It supports:
  + Automatic Deployment / Continuous Delivery
  + Autoscaling
  + Load balancing
  + Logs & metrics



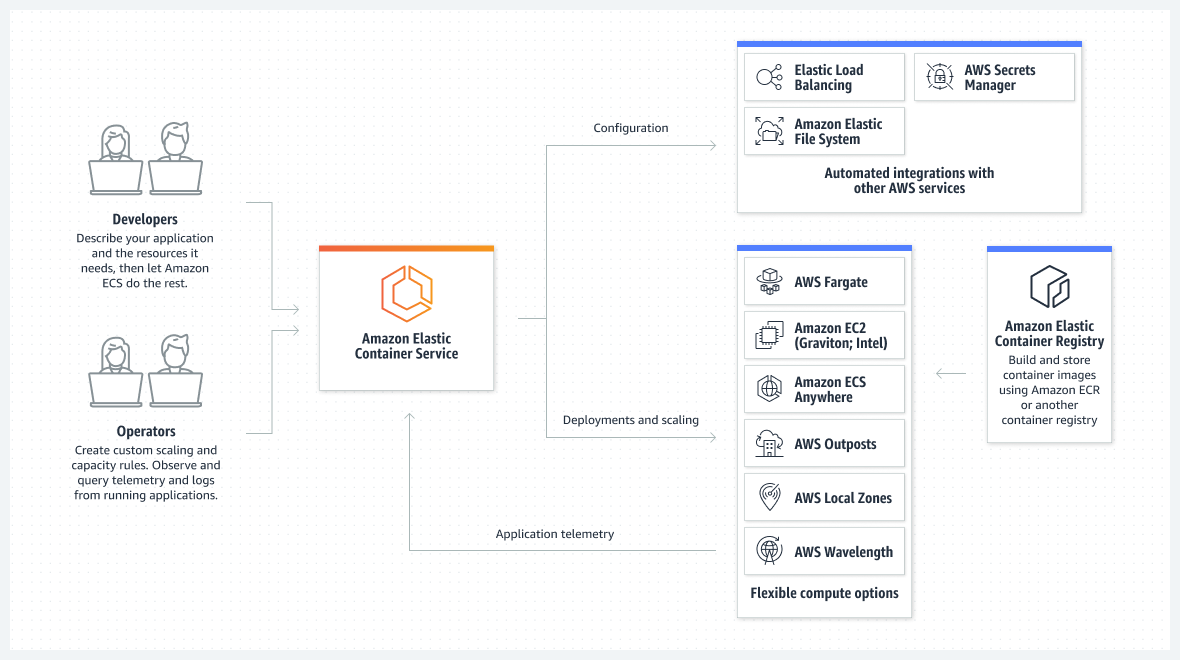
AWS **App2Container (A2C)** is a command line tool to help you lift and shift applications that run in your on-premises data centers or on virtual machines, so that they run in containers that are managed by Amazon ECS, Amazon EKS, or AWS App Runner.



***Amazon ECS***

Amazon Elastic Container Service (ECS) is a fully managed container orchestration service that helps you to more efficiently deploy, manage, and scale containerized applications. It deeply integrates with the AWS environment to provide an easy-to-use solution for running container workloads in the cloud and on premises with advanced security features using Amazon ECS Anywhere.

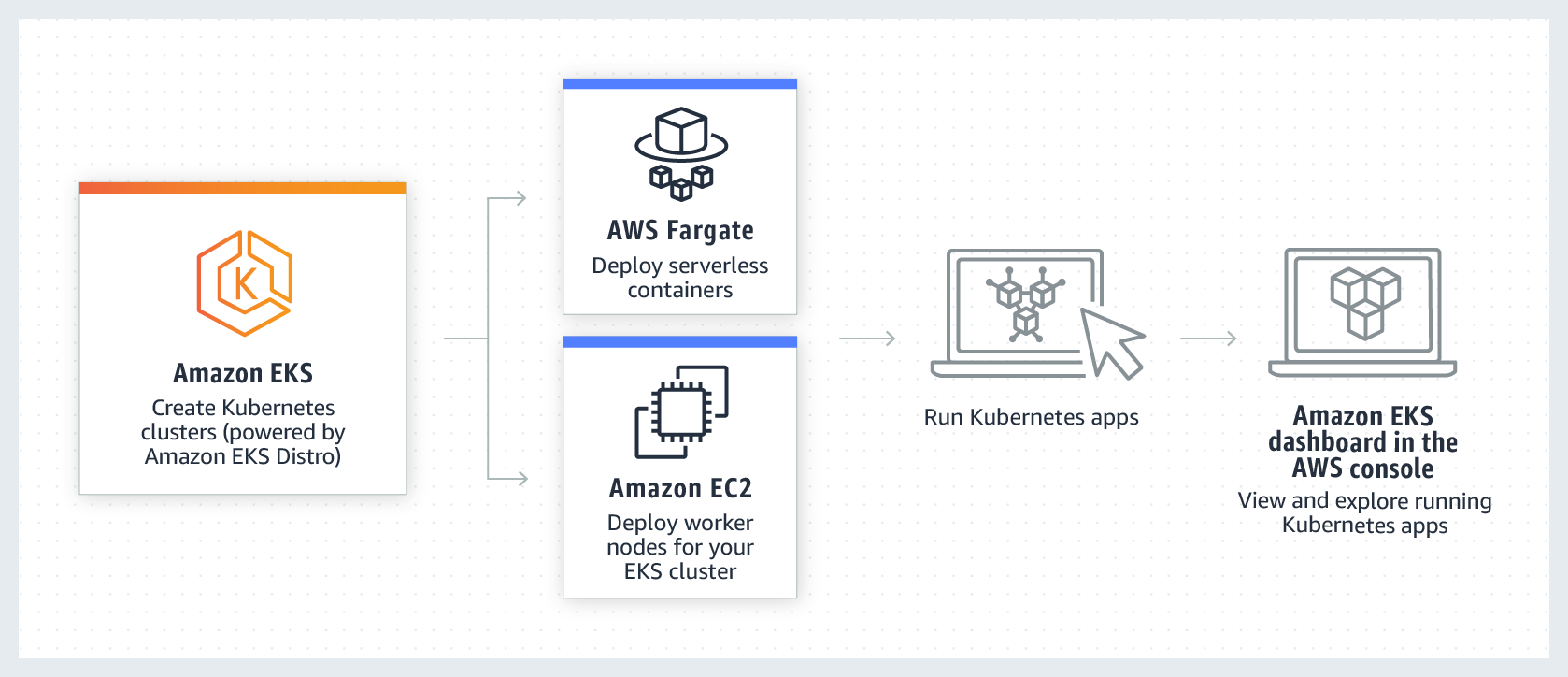
Simply describe your application and the resources required and Amazon ECS will launch, monitor, and scale your application across flexible compute options with automatic integrations to other supporting AWS services that your application needs. Perform system operations such as creating custom scaling and capacity rules, and observe and query data from application logs and telemetry.

[](https://aws.amazon.com/ecs/)

The diagram shows how Amazon ECS launches, monitors, and scales applications across flexible compute options with automatic integrations to other AWS services. Developers describe their applications and the resources it needs, and let Amazon ECS do the rest. Operators perform system operations such as creating custom scaling and capacity rules, and observe and query data from application logs and telemetry. Three sections display from left to right.

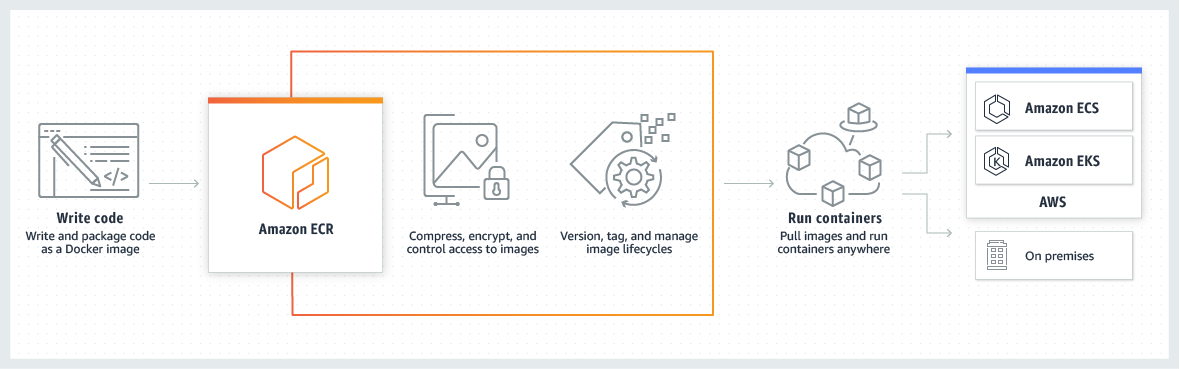
***Amazon EKS***

Amazon Elastic Kubernetes Service (Amazon EKS) is a managed Kubernetes service to run Kubernetes in the AWS cloud and on-premises data centers. In the cloud, Amazon EKS automatically manages the availability and scalability of the Kubernetes control plane nodes responsible for scheduling containers, managing application availability, storing cluster data, and other key tasks. With Amazon EKS, you can take advantage of all the performance, scale, reliability, and availability of AWS infrastructure, as well as integrations with AWS networking and security services. On-premises, EKS provides a consistent, fully-supported Kubernetes solution with integrated tooling and simple deployment to AWS Outposts, virtual machines, or bare metal servers.



***Amazon ECR***

Amazon Elastic Container Registry (Amazon ECR) is a fully managed container registry offering high-performance hosting, so you can reliably deploy application images and artifacts anywhere.

[](https://aws.amazon.com/ecr/)

***Docker vs Kubernetes***

While Docker is a container runtime, Kubernetes is a platform for running and managing containers from many container runtimes. Kubernetes supports numerous container runtimes including Docker, containerd, CRI-O, and any implementation of the Kubernetes CRI (Container Runtime Interface).

A good metaphor is Kubernetes as an “operating system” and Docker containers are “apps” that you install on the “operating system”.

***Containerizing ML application***

**Steps** :-

1. Clone the repo in AWS Cloud9 or any other IDE (GIthub workspace/ VS Code…)

GitHub repo:: <https://github.com/noahgift/Python-MLOps-Cookbook>

1. Create a ECR repo (public/private)
2. Use push commands of ECR to push the code from IDE to ECR to create image of the container.
3. Create AWS App runner from ECR image which will give you url end point for your web app.

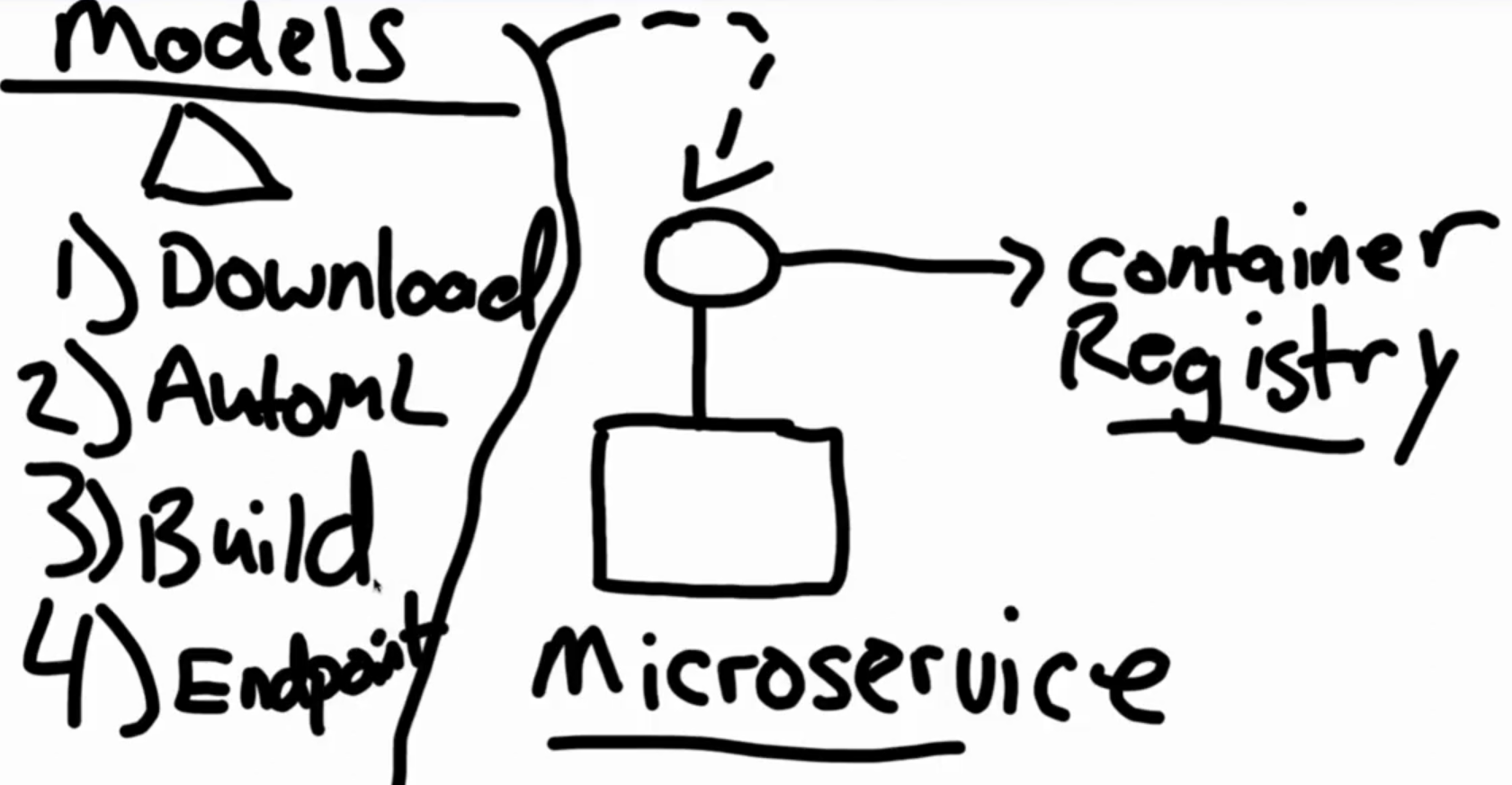
AWS App runner supports continuous delivery so it will build the code automatically once new changes pushed to ECR from IDE.

**Steps** :-

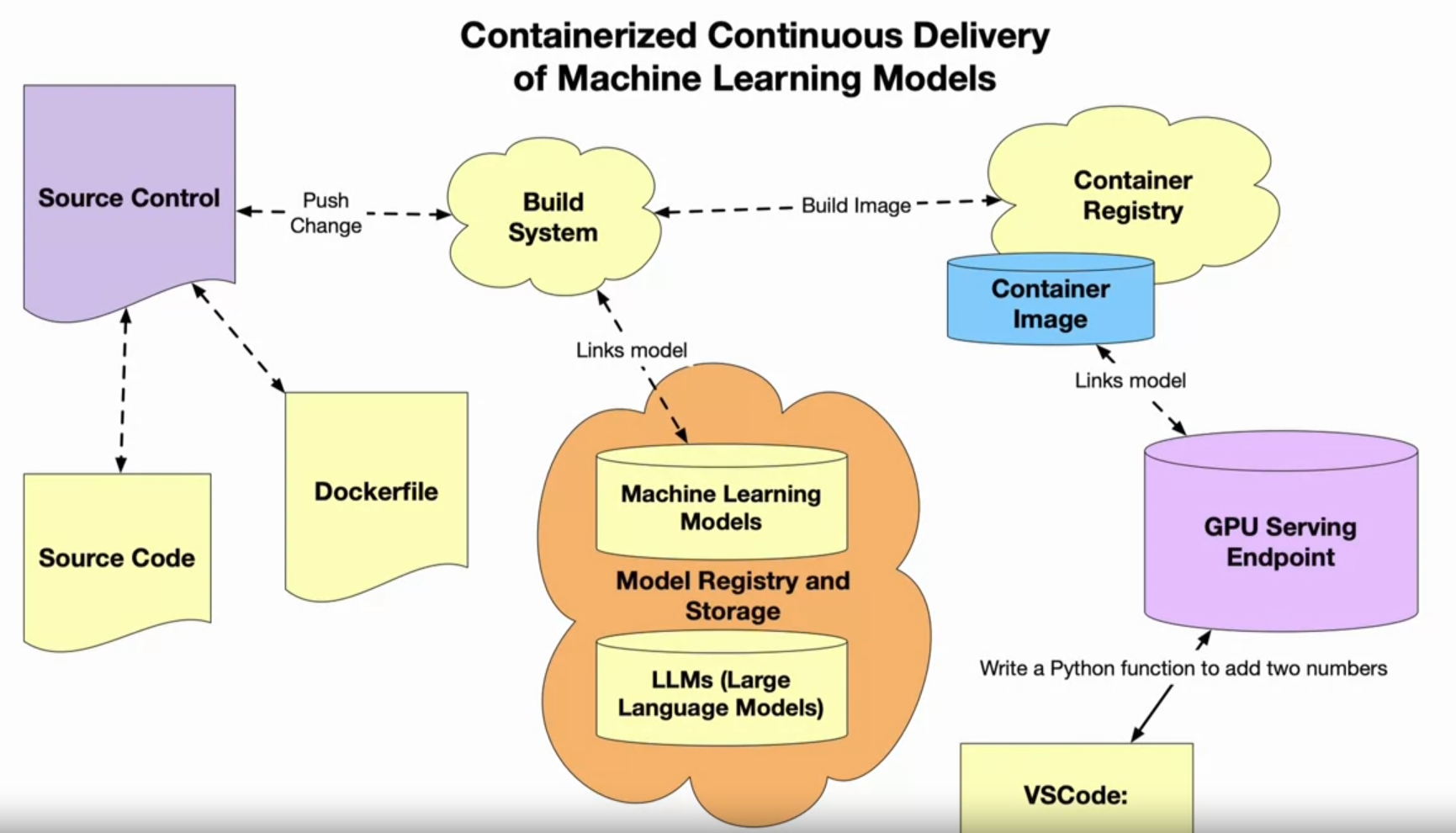
1. Clone the repo in AWS Cloud 9 or any other IDE

<https://github.com/noahgift/mlflow-project-best-practices>

1. Make all
2. Create a ECR repo (public/private)
3. # Use push commands of ECR to push the code from IDE to ECR to create image of the container.
4. Authenticate ecr repo: aws ecr get-login password <copy from ecr push commands>
5. Docker build -t <image name same as ECR repo name> <copy from ecr push commands>
6. Docker image ls
7. Docker run -p 0.0.0.0:8080 <docker image id of step 5>
8. Push to ecr :
   1. docker tag <ecr repo name>:latest <copy from ecr push command>
   2. docker push <copy from ecr push commands>



1. Download – Downloading models from hub like hugging face, ollama, tensorflow hub
2. AutoML – Databricks, Sagemaker Azure ml studio
3. Build – Building model from scratch
4. Endpoints



| **Title** | **Brief Description** |
| --- | --- |
| [Virtualization and Elasticity](https://paiml.com/docs/home/books/cloud-computing-for-data/chapter03-virtualization-containers-elasticity/) | A reading on virtualization and elasticity that provides a foundational overview of cloud computing concepts. |
| ["Distroless" Container Images](https://github.com/GoogleContainerTools/distroless) | The official GitHub Repo for distroless containers that reduce the size of container images. |
| [Example GitHub project with Rust distroless](https://github.com/alfredodeza/rust-distroless-azure) | A Rust example project that uses distroless container technology to deploy Rust to Azure. |
| [Creating a container image for use on Amazon ECS](https://docs.aws.amazon.com/AmazonECS/latest/developerguide/create-container-image.html) | Documentation showing how to build and use container images on ECS. |
| [Azure Functions with Rust](https://learn.microsoft.com/azure/azure-functions/create-first-function-vs-code-other?tabs=go%2Cmacos&WT.mc_id=academic-0000-alfredodeza) | An official Microsoft reading on how to use Azure Functions with Rust |
| [Introduction to Containers](https://docs.google.com/presentation/d/1uBlq4CMeQSffU3wwyU0xRrSR7buud20t/edit#slide=id.p1) | AWS official powerpoint presentation discussing container technology. |

**Summary of Lesson** This lesson explained containers for encapsulating reusable, reliable ML applications, leveraging continuous delivery pipelines, Kubernetes orchestration, infrastructure as code, and optimized distroless images.

**Top 3 Key Points**

* Containers enable portable ML applications
* Automated pipelines deploy containers
* Orchestration coordinates scale

