CONTAINERS AND ECS

Introduction to Containers

Another type of compute, container computing. Reminder, Virtualization is multiple O/S's operating on a single collection of hardware (guest O/S's). Containerization improves on Virtualization by aggregating and avoiding duplicating O/S's on different Virtual Machines in a system.

- With containerization, you have Host Hardware at the bottom, Host O/S on that, then a Container Engine (think Docker) atop the Host O/S. Now apps/whatever can run within isolated containers in the Container Engine. Each app no longer requires a full O/S, so they run lighter and allow more apps running on single piece of hardware
- Container is a running copy of a Docker Image, a Docker Container adds read/write capability to Docker Image. Similar to how an EC2 instance is a running copy of an EBS Volume. Docker Images are set up in layers which are read only, changes to layers are tracked using differential architecture (only tracks changes from 1 layer to another)
- Collections of Container Images are kep in a Container Registry Eg. Docker Hub
- Dockerfiles used to create Docker Images
- Great substitute for VMs if you don't need a full O/S for the container

DEMO - Creating 'container of cats' Docker Image

Create a docker image containing the 'container of cats' application.

- Prereq: Create a DockerHub account

1. 1 Click Deploy.
2. Nav to new EC2 instance > Connect > Session Manager
3. Commands https://learn-cantrill-labs.s3.amazonaws.com/awscoursedemos/0030-aws-associate-ec2docker/lesson_commands.txt

- `sudo amazon-linux-extras install docker` - Install Docker
- `sudo service docker start` - Start Docker
- `docker ps` - Test Docker. Expect a permissions error
- `sudo usermod -a -G docker ec2-user` - Add permissions
- `exit` and reopen a Session Manager
- `sudo su - ec2-user` - Log in as EC2 user
- `docker ps` now works
- `cd container`, `ls -la`
- `docker build -t containerofcats .` - Build container image
- `docker images --filter reference=containerofcats` - Show images in this container
- `docker run -t -i -p 80:80 containerofcats` - Map port 80 on container to port 80 on ec2 instance with cats image
- Log in to docker hub docker login --username=YOUR_USER docker images docker tag IMAGEID YOUR_USER/containerofcats docker push YOUR_USER/containerofcats:latest

ECS - Elastic Container Service - Concepts

ECS is to containers as EC2 is to virtual machines. A managed container-based compute service

- Runs in two modes 1. EC2 2. Fargate
- ECS let's you create a cluster. A Cluster is where you container runs from.
- Elastic Container Registry is the ECS Container Registry
- Container Definition: Defines Images and Ports
- Task Definition: A self contained application; the app as a whole. Stores a Task Role, an IAM Role.
- Task Role: IAM Role which the TASK assumes

EXAM: Task Roles are the best practice way giving containers within ECS permission to interact with AWS Resources

- ECS Service. Config'd with Service Definition: Tells how a task will scale and determines high availability. Can use load balancer. Restarts. EXAM: Cluster Modes available within ECS: Network Only (Fargate), EC2 Linux + Networking, EC2 Windows + Networking EXAM: Benefits of containers: Fast to start up, Portable, Lightweight

ECS - Cluster Mode

ECS is capable of running in EC2 mode or Fargate mode. EC2 mode deploys EC2 instances into your AWS account which can be used to deploy tasks and services.

- With EC2 mode you pay for the EC2 instances regardless of container usage.
- Fargate mode uses shared AWS infrastructure, and ENI's which are injected into your VPC.
- You pay only for container resources used while they are running.

The Two Modes when running ECS: EC2 Mode, Fargate Mode

- Main differentiator: What you manage VS what AWS manages

EC2 Mode

- Creates EC2 instances as Container Hosts; You'll be paying for these Hosts/Instances
- EC2 Mode for when you want to use containers in your infrastructure but you NEED to manage container host capacity and availability

Fargate Mode

- No management of EC2 instances as Container Hosts; no servers to manage.
- Not paying for EC2 instances.
- The main difference is how containers are Hosted: Fargate Shared Infrastructure instead of EC2 instances. -- Tasks / Services run from the Shared Insfrastructure and are injected in
- Only pay for the container Resources that you consume

EXAM: When to use EC2 vs ECS (EC2) vs Fargate

- EC2 VS ECS. Containers? ECS. Containers great for isolating applications
- EC2? Large consistent workload, price conscious, make use of existing reservations
- Overhead conscious? Fargate. Less mgmt overhead.
- Small / Burst / Batch / Periodic workloads? Fargate (since you only pay for consumed resources)

DEMO ECS - Deploying 'container of cats' using Fargate

Create a Fargate Cluster, create a task and container definition and deploy the world renowned 'container of cats' Application from Dockerhub into Fargate.

1. Create Cluster: ECS Console > Clusters > Create Cluster > Networking Only option > name "allthecats", (don't tick New VPC box, use Default VPC instead by not clicking) > Create. If you get ECS error, redo these steps
2. Create Task Definition (deploy container): Task Definitions > New > Compatibility: Fargate > Next Step > def name "containerofcats", no task role needed, operating system family: linux, task size: 1bg memory, 0.5vCPU > click "Add Container": name "containerofcatsweb", image "docker.io/acantril/containerofcats", Memory Limit Soft 1024, Port Mappings: 80 tcp > Add
3. Run it: Clusters > allthecats > Tasks tab > Launch type: fargate, OS family: linux, select VPC: Cluster VPC: default selection, subnet: select 2 at random > Create/Add
4. Cleanup > Stop Task Definition > Task: Actions: Deregister > Cluster: allthecats: Delete

ECR - Elastic Container Registry

Like Docker Hub but the AWS version. A managed Container Image Registry service

- Each AWS account has a public and private registry and can have many repositories -- Public = Public Read only, R/W requires permissions -- Private = Permissions required for any R/O or R/W
- In each repo you can have many container images
- Images can have several tags

Benefits of ECR

- Integreated with IAM for Permissions
- Offers security scannings, Basic and Enhanced Scanning (enh. uses Inspector product)
- Near real-time metrics delivered into CloudWatch
- Logs API actions into CloudTrail
- Generates events deliverd to EventBridge
- Offers Replication (cross region and cross account)

Kubernetes - K8

Open source container orchestration system. Cloud agnostic product - can use on many platforms

- Cluster: Highly available cluster of compute resources -- Vocab: Cluster Control Plane, Cluster Nodes, containerd, kubelet, Kubernetes API -- The control plane orchestrates containerized applications which run on nodes
- Pods: Smallest unit of computing in K8's, often seeing one-container-one-pod. Pods are NON-PERMANENT; view them as temporary things that are created for job and gone when job is done
- Cluster: A deployment of K8's
- Node: Resources within cluster; pods are placed on nodes to run
- Pods: Smallest unit in K8s, often 1 container to 1 pod
- Service; Abstraction running on 1 or more pods; what you'll understand as an application
- Job: ad-hoc, creates one or more pods until completion
- Ingress: How something external to the cluster can access the service

- Ingress Controller: Used to provide ingress (Eg. AWS Load Balance Controller) Default storage in K8s is ephemeral by default, provided locally by a node. Like ISVs on EC2
- Persistent Storage (PV) : Volume whose lifecycle lives beyond any 1 pod using it

Elastic Kubernetes Service (EKS)

Amazon Elastic Kubernetes Service (Amazon EKS) is a fully-managed, Kubernetes implementation that simplifies the process of building, securing, operating, and maintaining Kubernetes clusters on AWS.

- open-source and cloud agnostic
- K8s Control Plane scales and runs on multiple AZs
- Integrates with AWS svc's: ECR, Elastic Load Balancer, IAM, VPC
- EKS Cluster = EKS Control Plane & EKS Nodes
- etcd: the key-value store that k8's uses, is distributed across multiple AZs
- Nodes: can be Self-Managed (ec2 instance), Managed Node Groups (ec2 instance), or Fargate pods
- For Persistent Storage, EKS can use EBS, EFS, FSx Lustre, FSx for NetApp ONTAP

EKS Resources

- https://docs.aws.amazon.com/eks/latest/userguide/eks-compute.html
- https://docs.aws.amazon.com/eks/latest/userguide/fargate.html