aws COMMUNITY DAY

AOTEAROA

"I have a container image. Now what?"

TERENCE WHITE | 2025-09-18

aws COMMUNITY DAY AOTEAROA

About me

- Based in Wellington
- 30+ years in IT, 8+ years AWS experience
- Both Pro certs*, 2x Specialty certs*
- Community Builder since Nov 2021
- Committee of AWS User Group of NZ/A

Random Quote

– "I do servers and I do serverless.
I don't do something halfway in-between like Docker"

The situation

The situation

Hi,
I am his boss

The situation

The situation

The situation

The concerns

- Proserv company
  - No container running platform
  - No internal Ops/DevOps teams
  - No internal product owner
- I don't want to feed and water this
- Proof of Concept becomes Production
- APN partner so let's do this on AWS

AWS options with Docker

- ECS - Elastic Container Service
- EKS - Elastic Kubernetes Service
- ROSA - Red Hat OpenShift Service on AWS (managed Kubernetes)
- LightSail - easy-to-use cost-effective containers
- Elastic Beanstalk - run Docker containers on EC2 instances
- App Runner - turn a container image into a scalable web application
- Batch - schedules and executes containers using ECS or EKS
- Lambda - can run containers on events
- CodeBuild - can run custom Docker images on schedule
- Docker on EC2 - old school!

The product

- Free Open Source and paid Enterprise version
  - Guess which one we are looking at

- Docker compose
  - Postgres DB -> RDS
  - App is 1 container image
  - with many env variables
  - 1 persistent storage volume

- So ALB -> (1x app) -> RDS DB

Non-contender #1

Docker on EC2

- Manual set up and maintenance
  - including the persistent storage
- Manual ALB integration
- No health checks or self healing
- No external observability

Suitable for a once-off quick-and-dirty get it done now

Non-contender #2

Lambda Docker

- Event based rather than continuous running
- Container must implement Lambda Runtime API
  - no control over Docker image
  - not designed to be event based
- 15 minute max runtime

Suitable for larger Lambda functions or custom runtimes

Non-contender #3

Batch

- Batch processing through queues rather than continuous running
- Containers should start and run to a finish
  - "incompatible with interactive processing"
  - this image runs a web server and listens forever
- Can schedule on Spot and save money
- Runs containers on EKS, ECS EC2 or Fargate

Suitable for running non-interactive containers

Non-contender #4

CodeBuild

- 36 hour maximum container lifetime
- Can run custom Docker image from external registry
  - but overrides the ENTRYPOINT for custom Docker images
- Can specify no source and no artifacts
- Can set up to run in a VPC and can set incoming rules on SGs
  - integration with ALB will be interesting

Suitable for building software rather than continuous running

Non-contender #5

ROSA

- ~~OpenShift~~ Kubernetes managed by RedHat in your AWS account
- Needs relationship with RedHat
- Supported by both RedHat and AWS, billed by AWS
- Installation, management, maintenance and upgrades are performed by Red Hat SREs
- Minimum of 7 servers (3 control plane, 2 infra, 2 worker)

Suitable for orgs already running OpenShift elsewhere

Non-contender #6

Lightsail Containers

- Less cost, less options and less control
- Can't add persistent storage to containers
- Can't easily link to RDS database or ALB in VPC
  - need to do VPC peering
- Containers billed if disabled rather than deleted

Suitable for people optimising cost over absolutely all else

aws COMMUNITY DAY AOTEAROA

Non-contender #7

App Runner

- Fast, simple, and cost-effective way to deploy from source code or a container image directly to a scalable and secure web application …
- … if your container image is stored in ECR or ECR Public
  - ours is stored in Docker Hub
- No persistent storage

Suitable for devs building microservices in AWS

Non-contender #8

Elastic Beanstalk

- Can run containers on either
  - Docker on EC2 servers (recommended)
  - ECS platform (legacy)
- Can use images from DockerHub
- Hard to configure through Terraform
- No persistent storage

Suitable for teams lacking ECS/K8S skills

So what's left?

- ECS - Elastic Container Service
- EKS - Elastic Kubernetes Service
- ~~ROSA - Red Hat OpenShift Service on AWS (managed Kubernetes)~~
- ~~LightSail - easy-to-use cost-effective containers~~
- ~~Elastic Beanstalk - run Docker containers on EC2 instances~~
- ~~App Runner - turn a container image into a scalable web application~~
- ~~Batch - schedules and executes containers using ECS or EKS~~
- ~~Lambda - can run specially formatted containers on events~~
- ~~CodeBuild - can run custom Docker images~~
- ~~Docker on EC2 - old school!~~

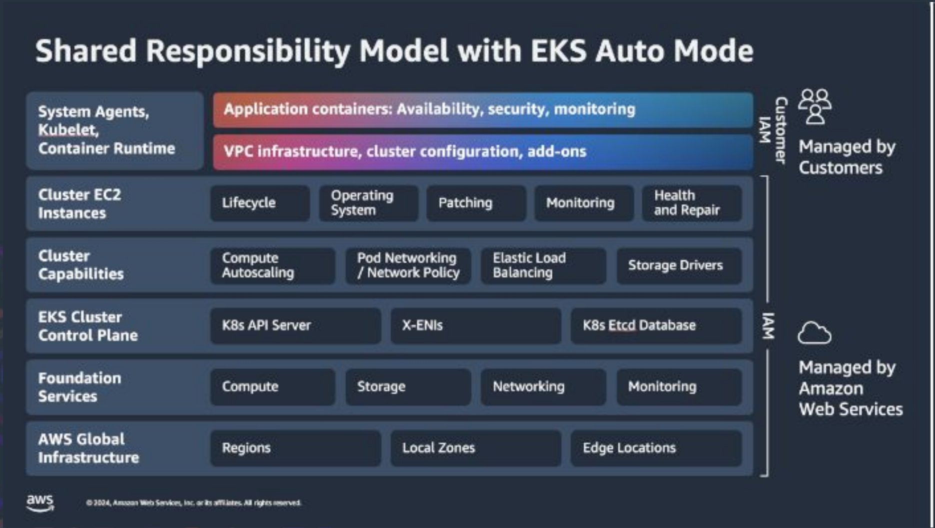
Contenders

	ECS	EKS
Orchestration?	Yes	Yes
Need to worry about servers?	No	No
Observability through CloudWatch?	Yes	Yes
ALB and RDS and Storage?	Yes	Yes
Terraform Provider?	AWS	AWS, Kubernetes, Helm

Contenders

	ECS	EKS (Auto Mode)
Orchestration?	Yes	Yes
Need to worry about servers?	No	No
Observability through CloudWatch?	Yes	Yes
ALB and RDS and Storage?	Yes	Yes
Terraform Provider?	AWS	AWS, Kubernetes, Helm

EKS Auto Mode



aws COMMUNITY DAY AOTEAROA


ECS
EKS
imgflip.com
JAKE-CLARK.TUMBLR

aws
COMMUNITY DAY
AOTEAROA

HELM

The package manager for Kubernetes

Helm is the best way to find, share, and use software built for Kubernetes.

```
resource "aws_eks_cluster" "eks" {
  name = "eks-cluster-${var.prefix}"

  access_config {
    authentication_mode = "API"
  }

  role_arn = aws_iam_role.eks-cluster.arn
  version  = "1.33"

  bootstrap_self_managed_addons = false

  compute_config {
    enabled       = true
    node_pools    = ["general-purpose"]
    node_role_arn = aws_iam_role.eks-node.arn
  }

  kubernetes_network_config {
    elastic_load_balancing {
      enabled = true
    }
    service_ipv4_cidr = "172.16.0.0/12"
    ip_family         = "ipv4"
  }

  storage_config {
    block_storage {
      enabled = true
    }
  }
```

```
  vpc_config {
    endpoint_private_access = true # for the nodes to communicate with the control plane
    endpoint_public_access  = true # for me to communicate with the control plane.
    public_access_cidrs     = [for s in local.ip-addresses-allowed : "${s}/32"]
    subnet_ids              = aws_subnet.private[*].id
  }

  upgrade_policy {
    support_type = "STANDARD"
  }

  zonal_shift_config {
    enabled = true
  }

  #  enabled_cluster_log_types = [ "api","audit","authenticator","controllerManager","scheduler" ]
  enabled_cluster_log_types = []

  # Ensure that IAM Role permissions are created before and deleted
  # after EKS Cluster handling. Otherwise, EKS will not be able to
  # properly delete EKS managed EC2 infrastructure such as Security Groups.
  depends_on = [
    aws_iam_role_policy_attachment.eks-cluster_AmazonEKSClusterPolicy,
    aws_iam_role_policy_attachment.eks-cluster_AmazonEKSComputePolicy,
    aws_iam_role_policy_attachment.eks-cluster_AmazonEKSBlockStoragePolicy,
    aws_iam_role_policy_attachment.eks-cluster_AmazonEKSLoadBalancingPolicy,
    aws_iam_role_policy_attachment.eks-cluster_AmazonEKSNetworkingPolicy,
    # since a node will come up in the private subnet and will need to reach out to the internet
    aws_nat_gateway.nat,
    aws_route_table_association.private
  ]
}
```

```
resource "helm_release" "xyz" {

  name             = "xyz"
  repository       = "https://xyz.github.io/xyz-helm-charts"
  chart            = "xyz"
  namespace        = "xyz"
  create_namespace = false
  lint             = true

  values = [templatefile("./values.tftpl", {
    HOST     = aws_rds_cluster.aurora.endpoint
    PORT     = aws_rds_cluster.aurora.port
    DATABASE = aws_rds_cluster.aurora.database_name
    USER     = aws_rds_cluster.aurora.master_username
    PASSWORD = aws_rds_cluster.aurora.master_password
    VERSION  = "v0.2.0"
  })]
}
```

Turns out

— *"I still haven't done Docker"*

Feedback and Q&A

Go to
pigeonhole.at

Enter passcode

AWSCDNZ25

aws
COMMUNITY DAY
AOTEAROA

aws

COMMUNITY DAY

AOTEAROA