CON214

INTRODUCTION TO AWS FARGATE

Anthony Suarez – GM, Amazon ECS & ECR
Deepak Dayama – Senior Product Manager, Amazon ECS

November 29, 2017

DAY ONE!

BUILDING AN ECOSYSTEM

WHY DO WE LOVE CONTAINERS?

Packaging

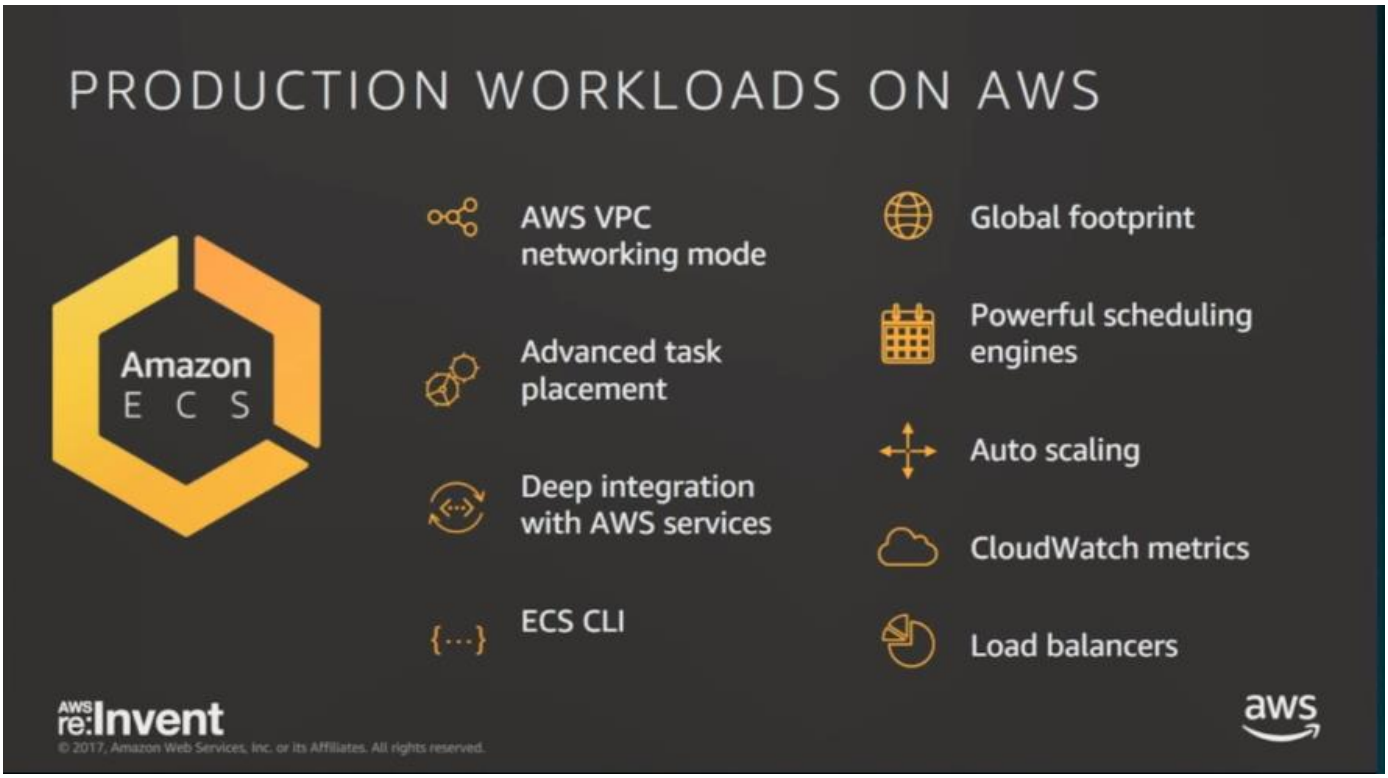
Distribution

Immutable infrastructure

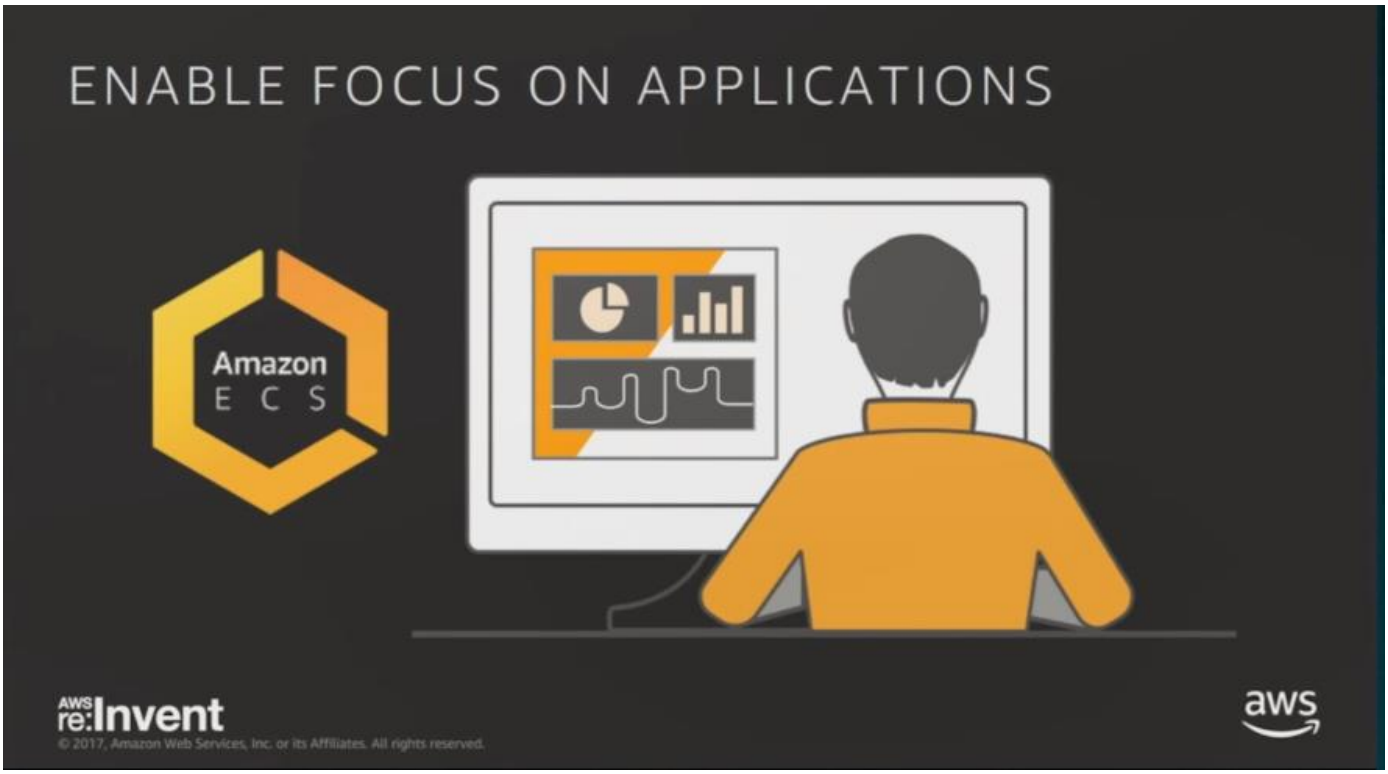
HELPING CUSTOMERS SCALE CONTAINERS

450+%
growth

Hundreds of millions
of containers started each week
millions
of container instances

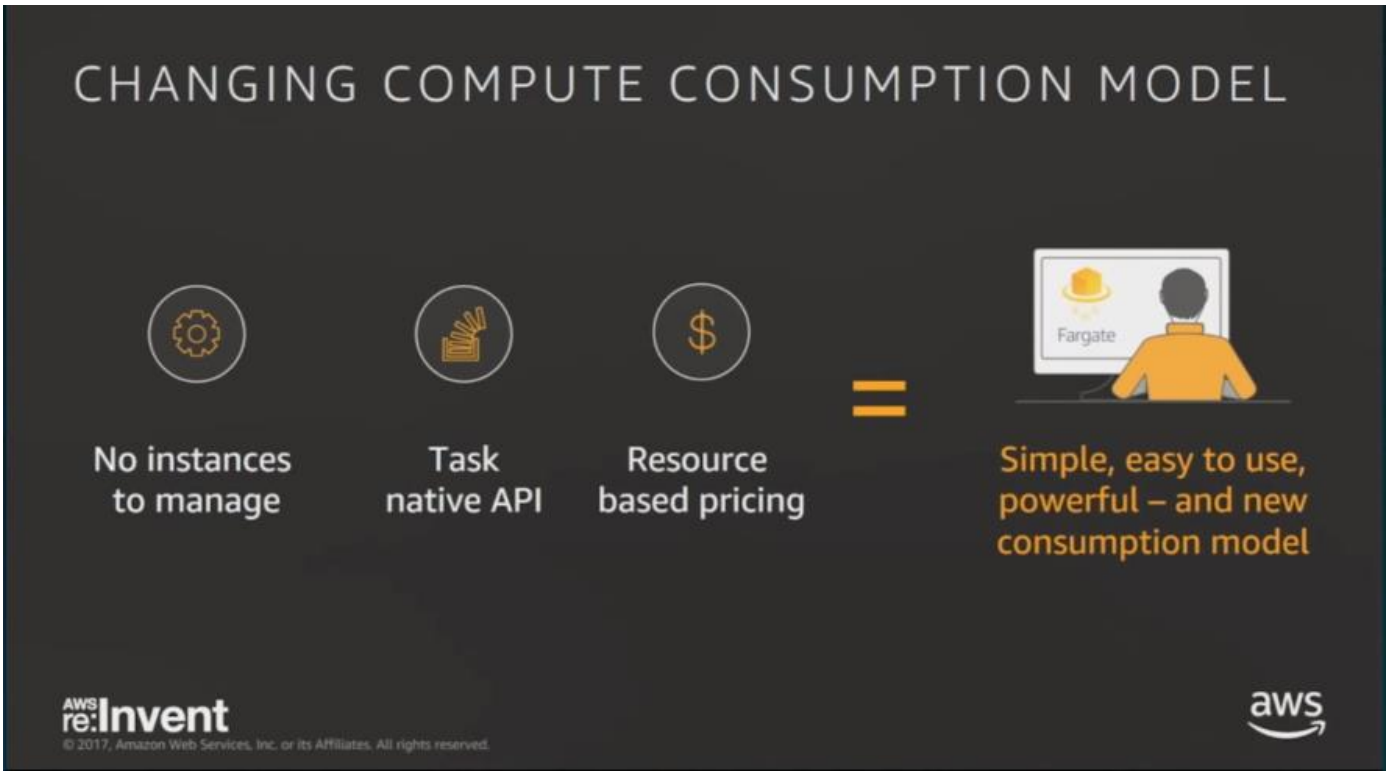


We gave you the same powers that you use when working with EC2 instances.

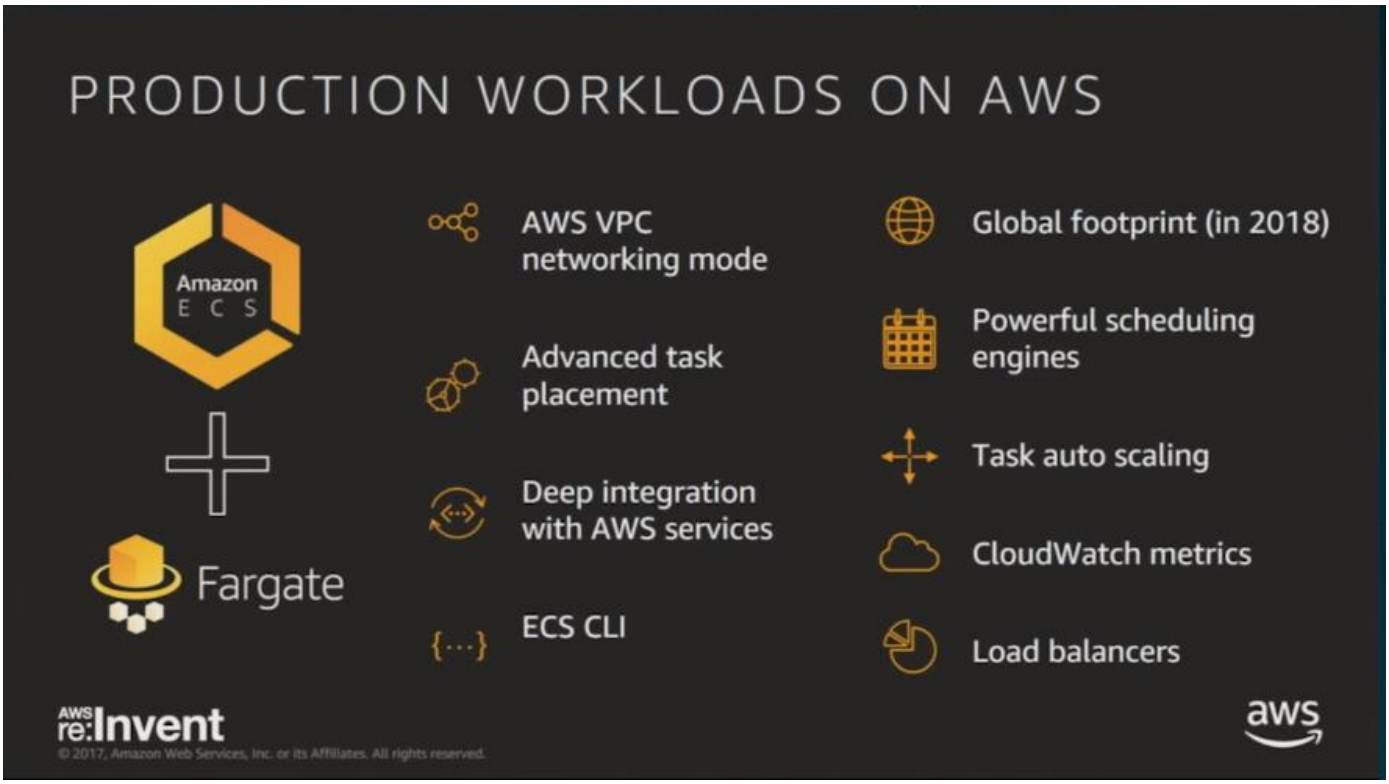


We want you to simply think about modelling your applications at the task level and leave the undifferentiated heavy lifting of running your clusters to AWS.

AWS Fargate gives you a new compute primitive to build your apps around using containers and tasks, the cluster is simply a construct that means nothing anymore.

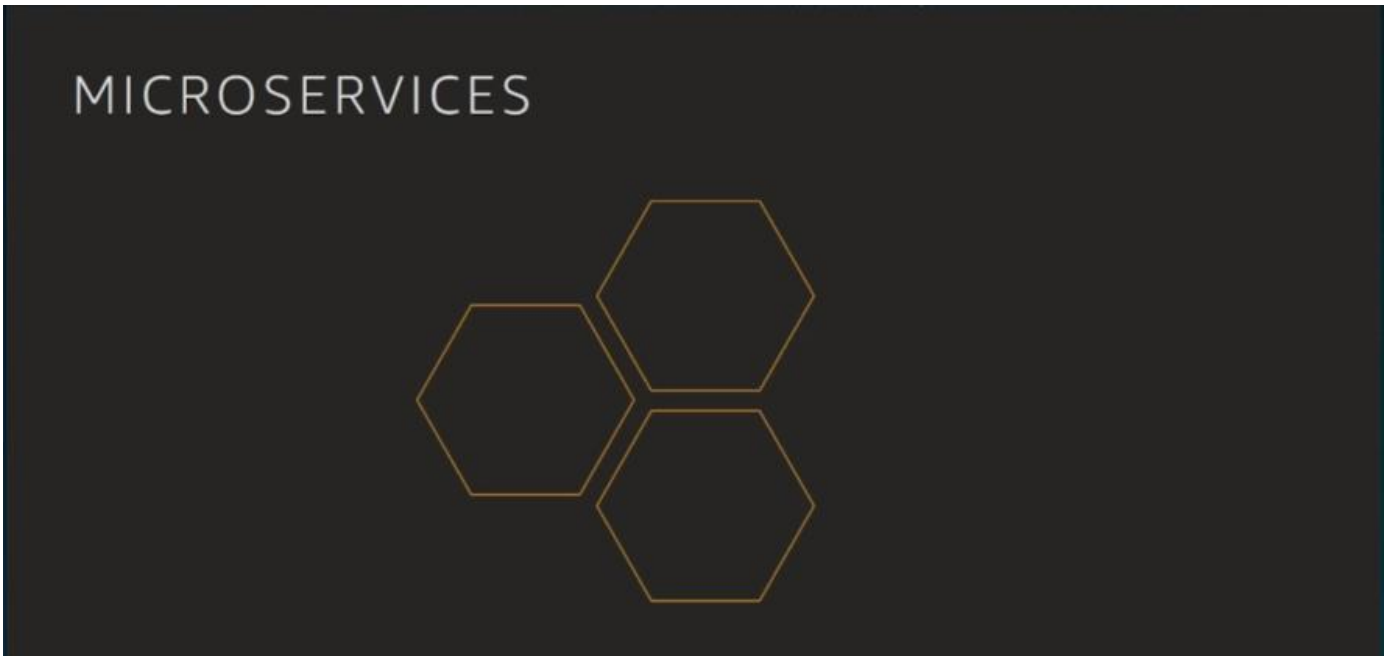


You simply pay for the CPU and memory that you provision on the per second level.



AWS Fargate is a technology that we built on top of ECS, you simply run Fargate tasks and it will be deployed for you.

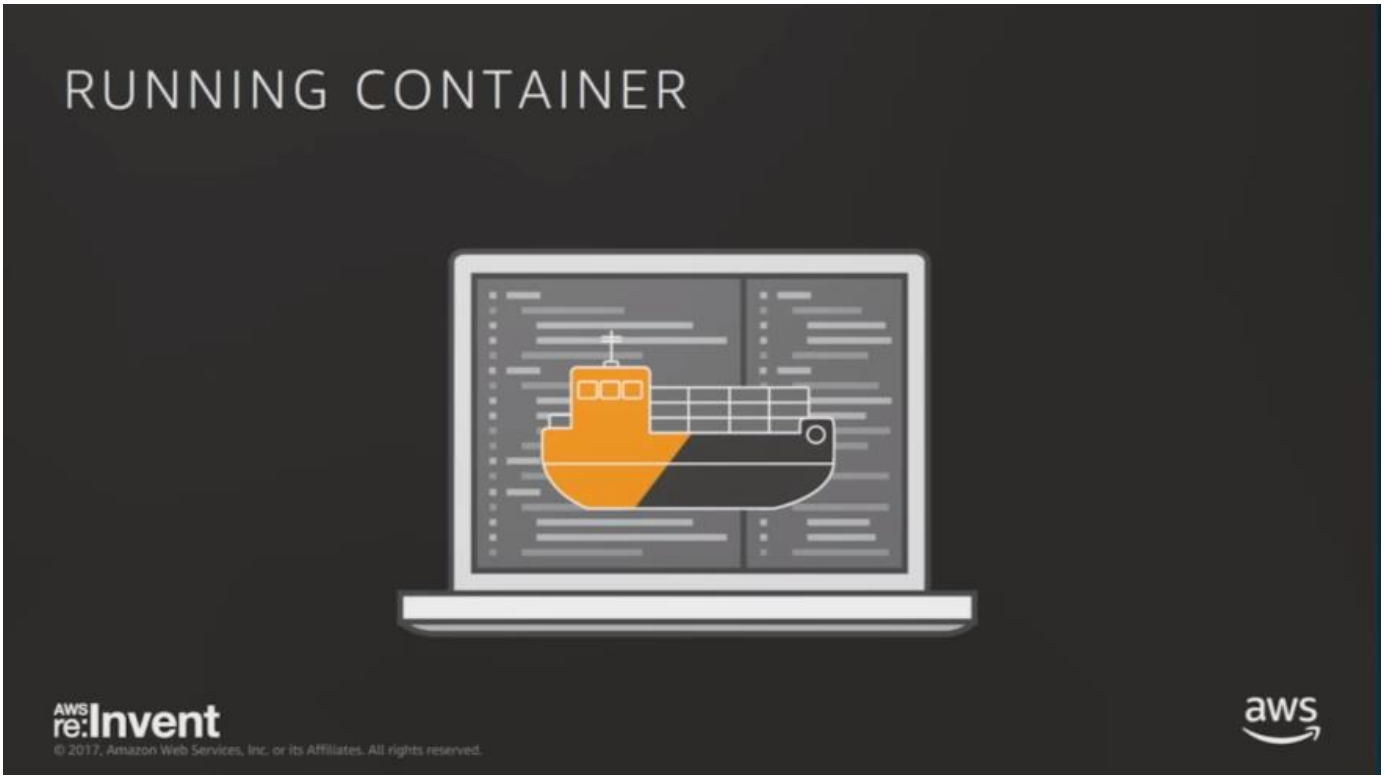
Amazon EKS will also be made to work with K8s clusters in 2018.



BATCH JOBS

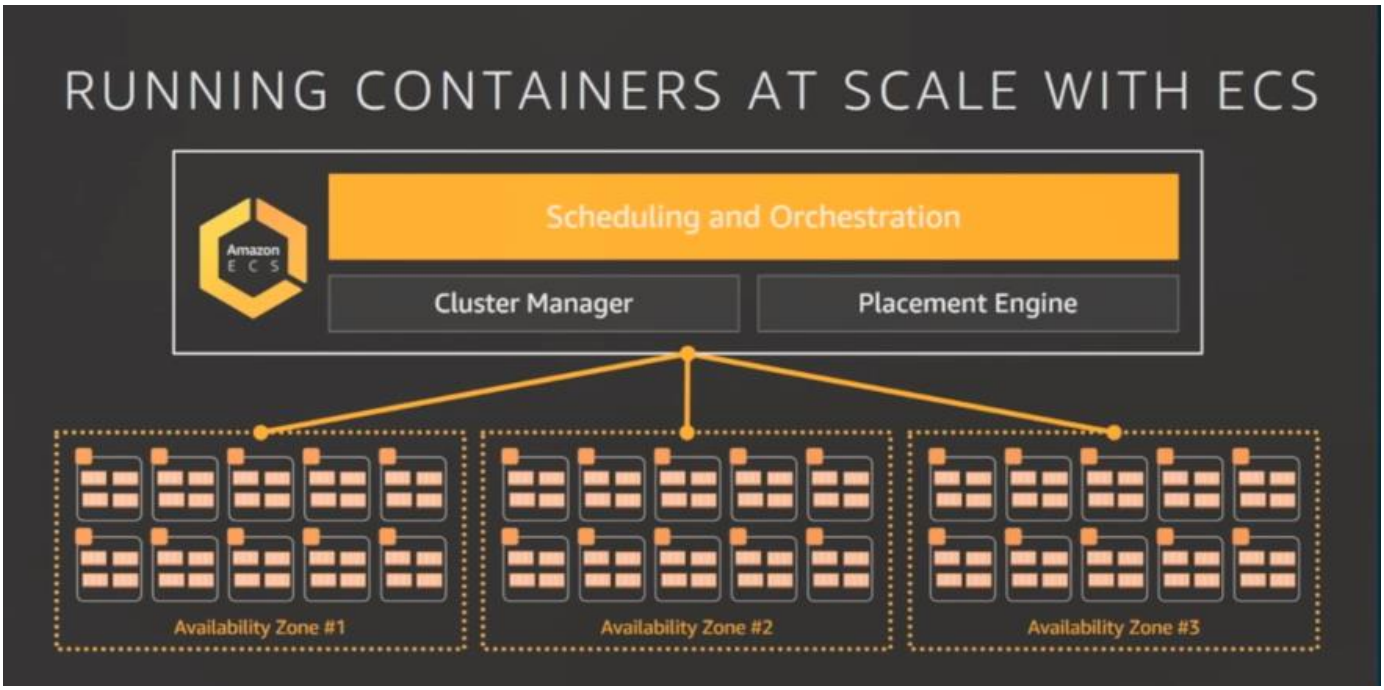
MIGRATION TO THE CLOUD

Let us now see how Fargate can be used in action,

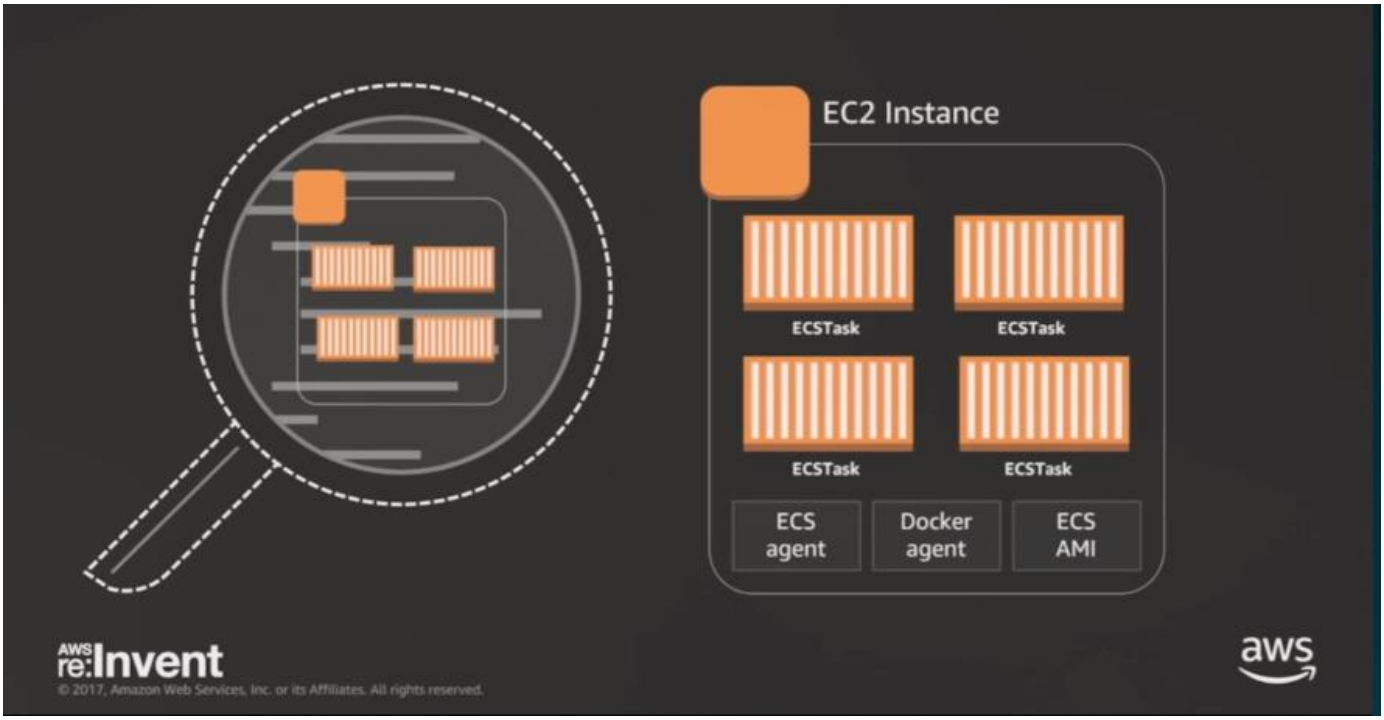


RUNNING CONTAINERS

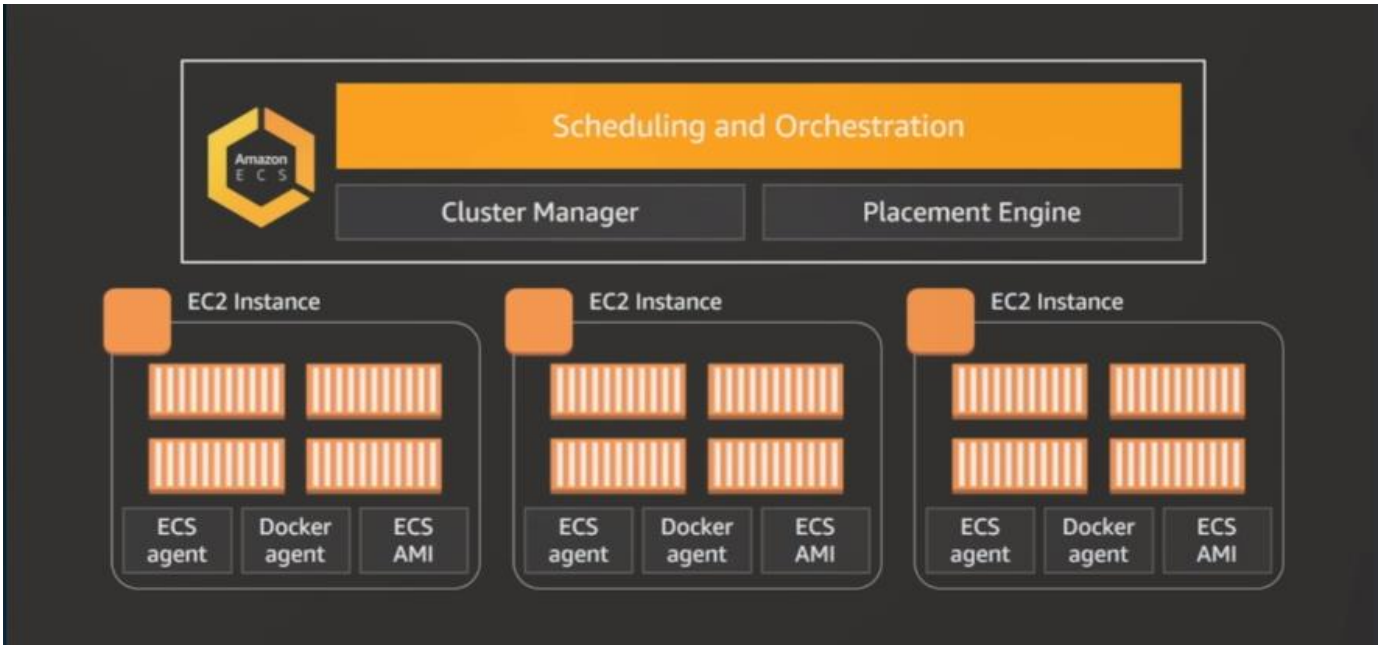
RUNNING CONTAINERS AT SCALE WITH ECS



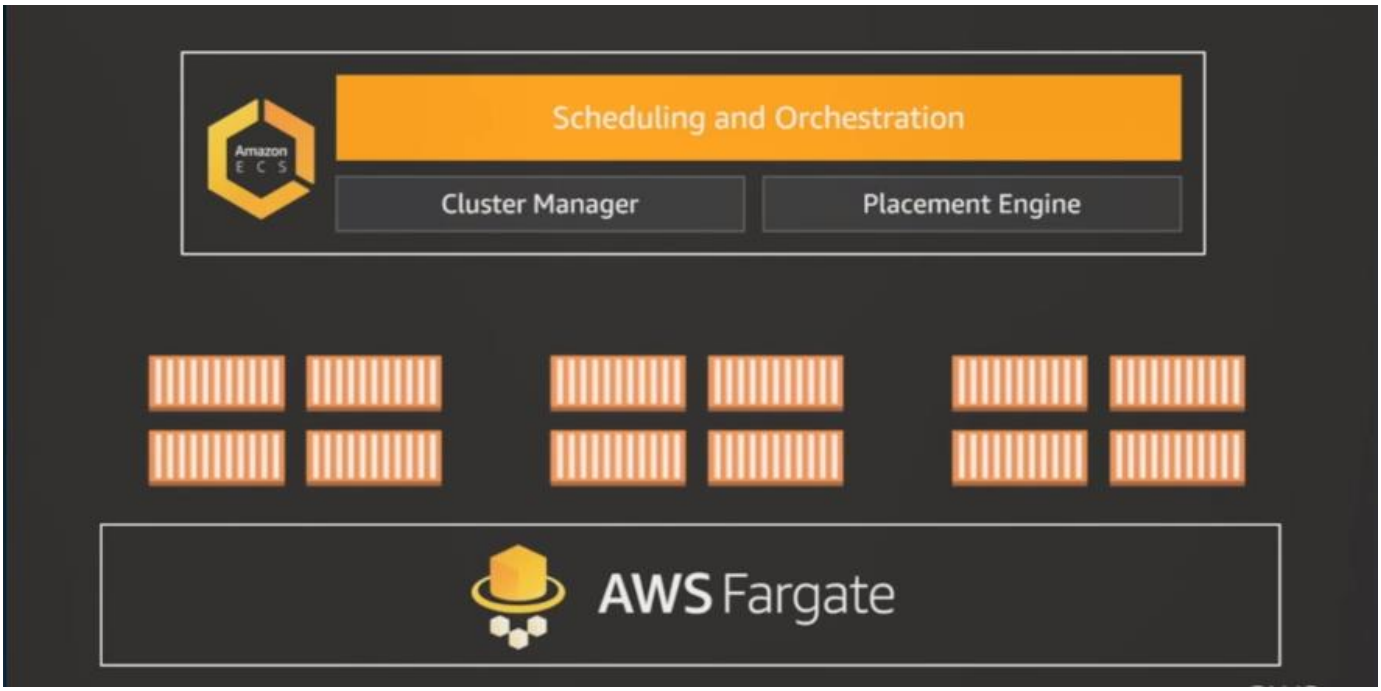
ECS allows you to be able to operationalize the scale using the cluster manage and placement emgine

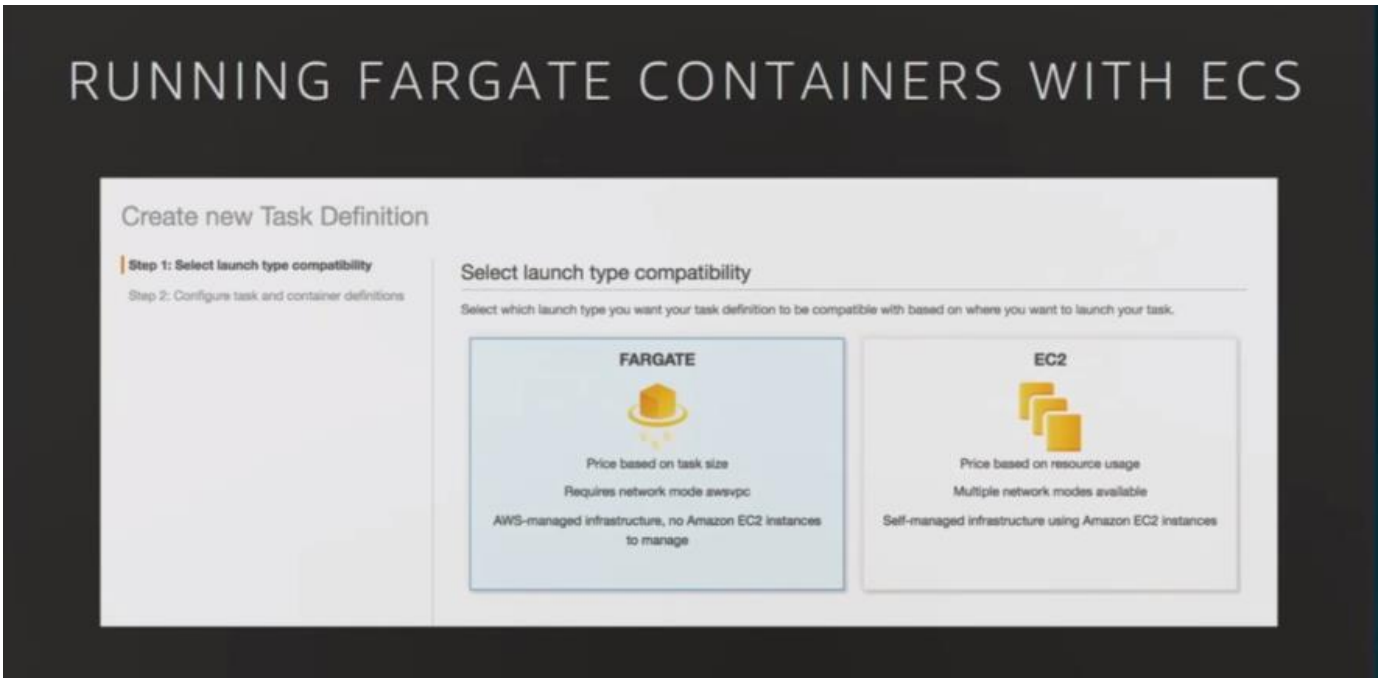


You also have to manage the Docker daemon and other things as above

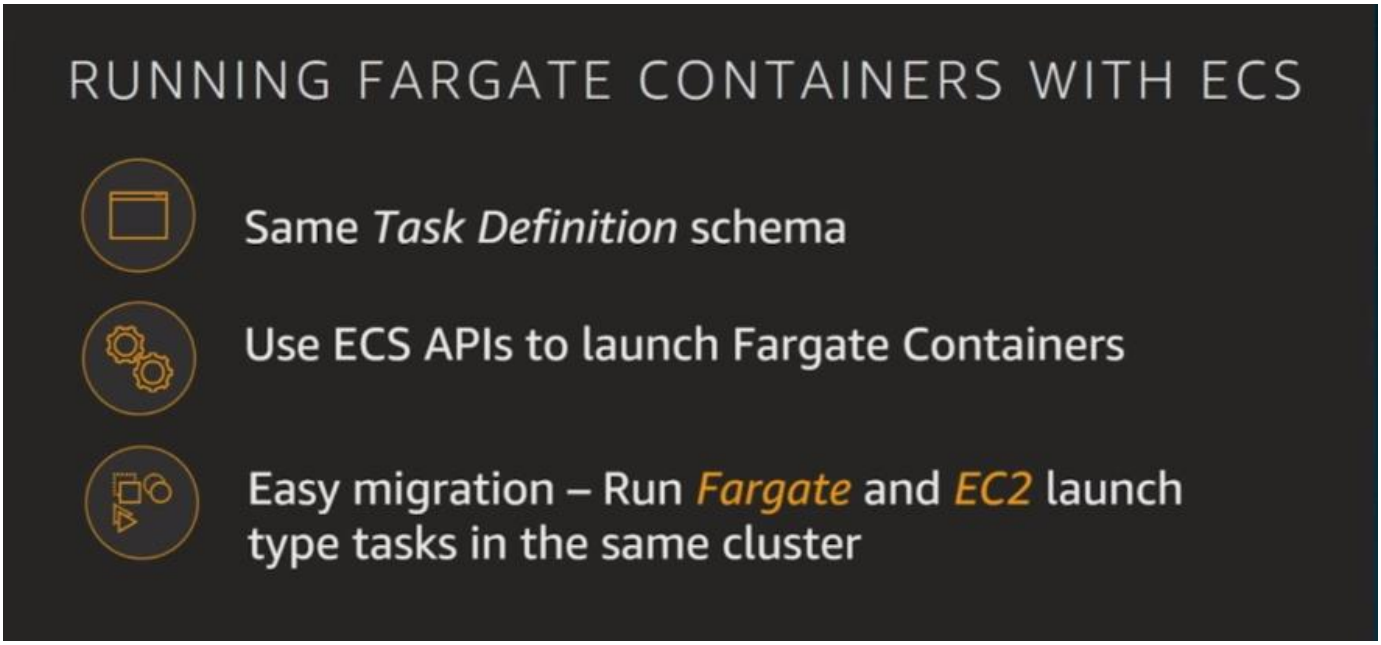


Fargate enables you to stop managing things above as below





Next, let us see how to use Fargate with ECS. When you want to launch an application with ECS, you now get a choice whether you want to launch it as a Fargate or EC2 launch type



```
0. Configure the ECS CLI
        ecs-cli configure –cluster tutorial –region us-east-1 \
                        –default-launch-type FARGATE –config-name tutorial
1. Setup cluster
        'ecs-cli up' #Create ECS cluster, VPC and two subnets by default
2. Create a compose file consisting of:
        - Wordpress
        - Set up CloudWatch Logs for container logs
        - Port binding
3. ECS Specific parameters in ecs-params.yml
        - Task size
        - Task Execution Role
        - Network Configuration for task placement

4. Monitor logs of your task
        - ecs-cli logs –-task-id <TASK_ID> –-follow
~
~
```

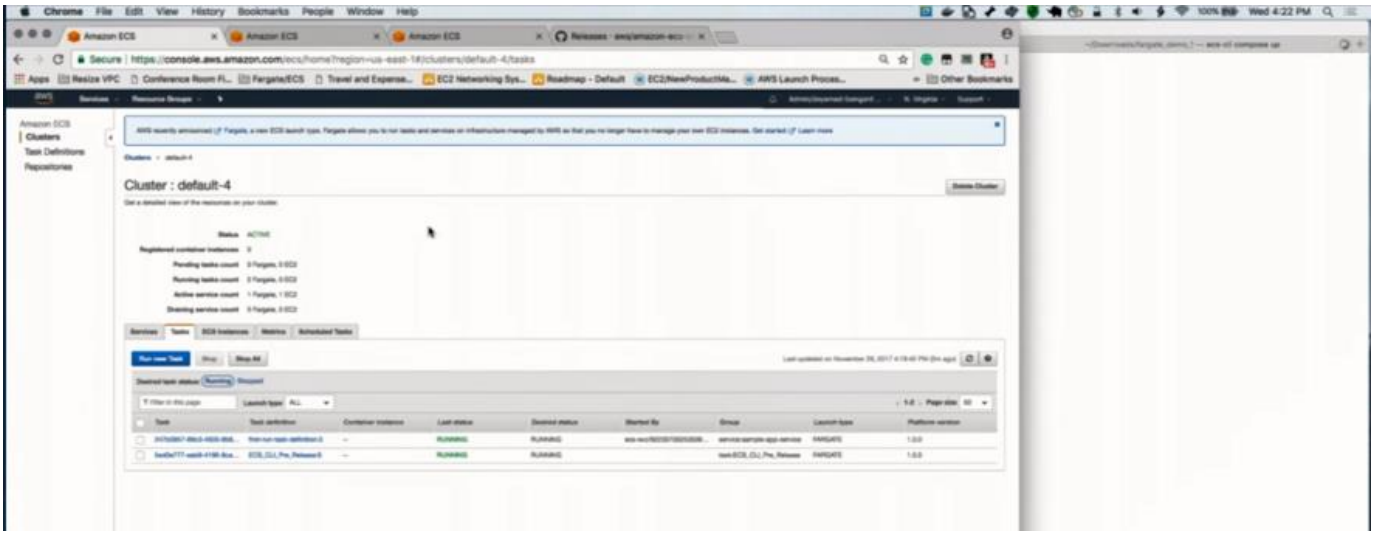
We will show the CLI experience using Fargate when using ECS

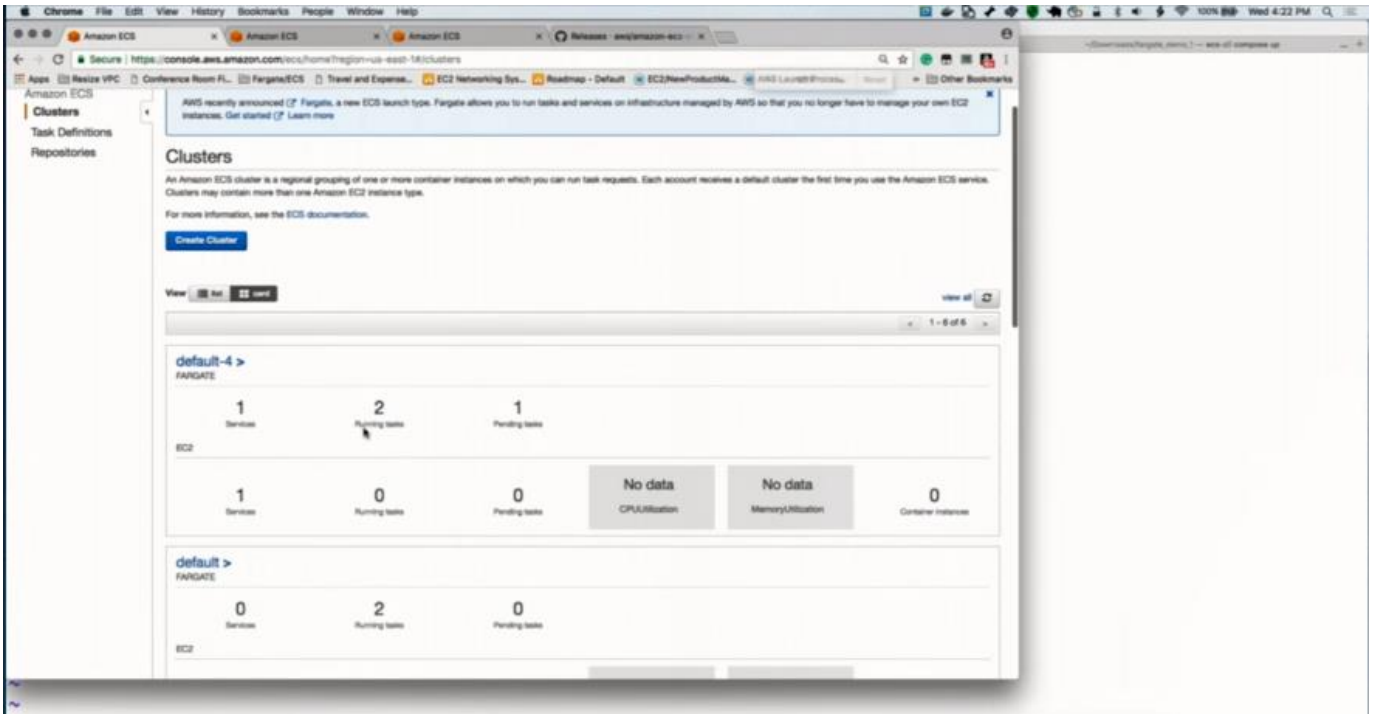
```
version: '2'
services:
  test:
    image: 424442929256.dkr.ecr.us-east-1.amazonaws.com/wordpress:latest
    ports:
      - "80:80"
    logging:
      driver: awslogs
      options:
        awslogs-group: awslogs-prefix
        awslogs-region: us-east-1
        awslogs-stream-prefix: wordpress
~
~
~
```

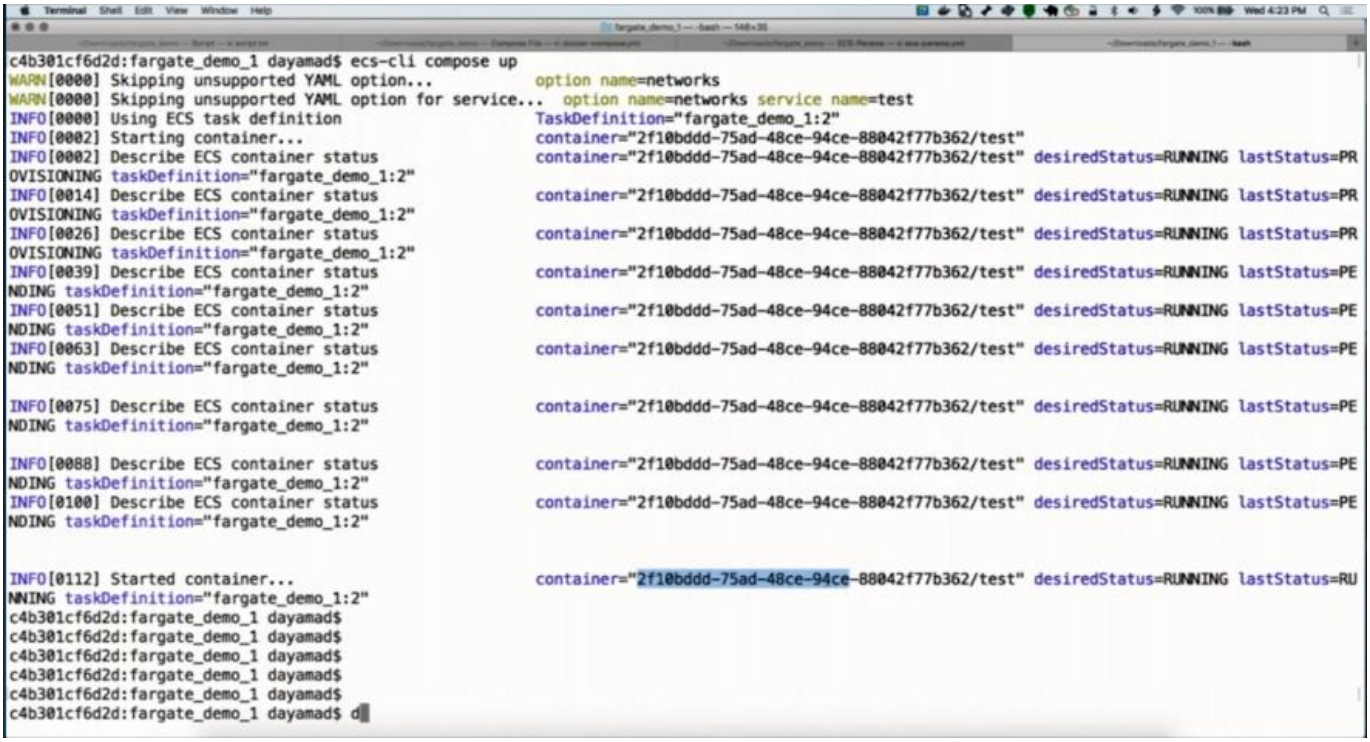
We also define our application as a compose file as above

```
c4b301cf6d2d:fargate_demo_1 dayamad$ pwd
c4b301cf6d2d:fargate_demo_1 dayamad$ ecs-cli compose up
WARN[0000] Skipping unsupported YAML option...          option name=networks
WARN[0000] Skipping unsupported YAML option for service...  option name=networks service name=test
INFO[0000] Using ECS task definition                    TaskDefinition="fargate_demo_1:2"
INFO[0002] Starting container...                        container="2f10bddd-75ad-48ce-94ce-88042f77b362/test"
INFO[0002] Describe ECS container status                container="2f10bddd-75ad-48ce-94ce-88042f77b362/test" desiredStatus=RUNNING lastStatus=PR
OVISIONING taskDefinition="fargate_demo_1:2"
```

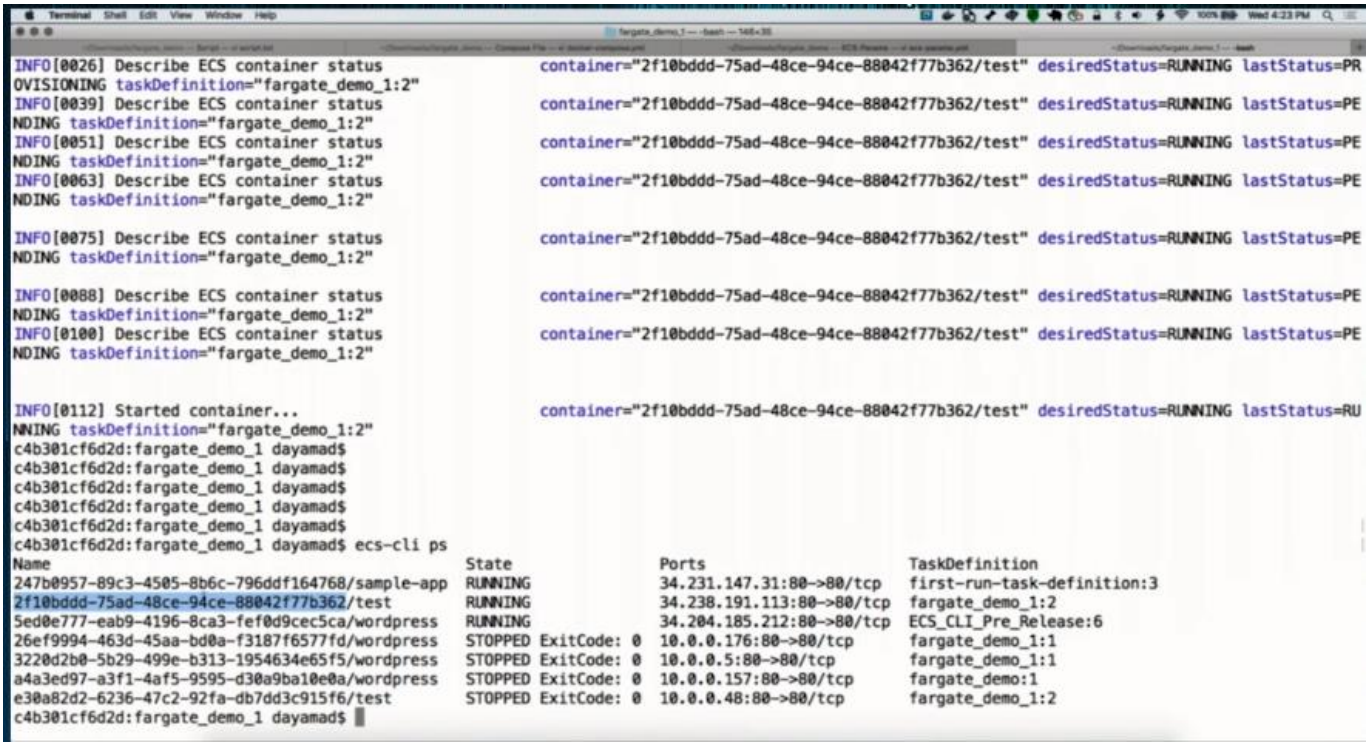
```
version: 1
task_definition:
  task_execution_role: arn:aws:iam::424442929256:role/ecsTaskExecutionRole
  ecs_network_mode: awsvpc
  task_size:
    mem_limit: 0.5GB
    cpu_limit: 256
run_params:
  network_configuration:
    awsvpc_configuration:
      subnets:
        - subnet-bc4b42b0
      assign_public_ip: ENABLED
      security_groups:
        - sg-341bb141
~
~
~
```


Cluster : default-4

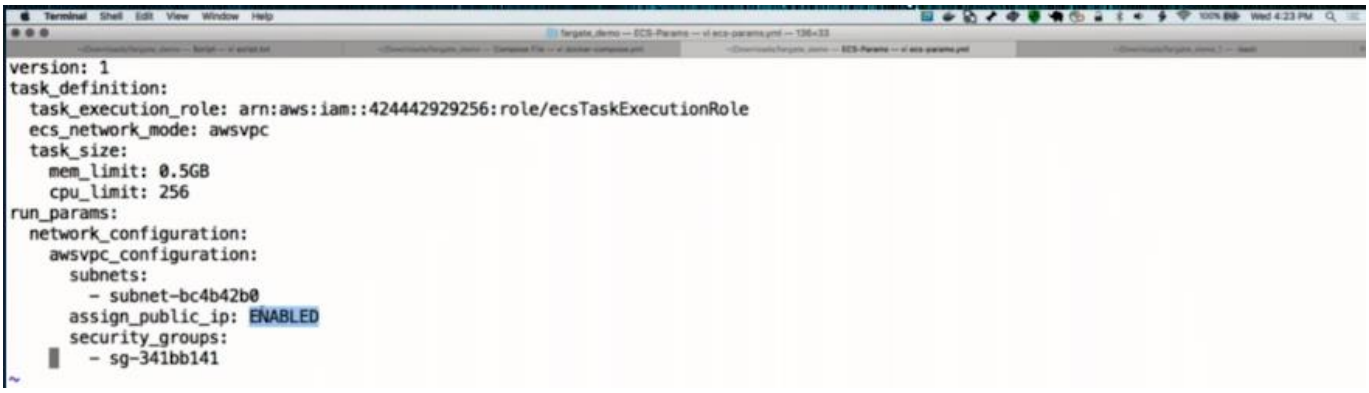

Clusters
Task Definitions
Repositories
Clusters
default-4 >
FARGATE
1
2
1
EC2
1
0
0
No data
No data
0
default >
FARGATE
0
2
0
EC2



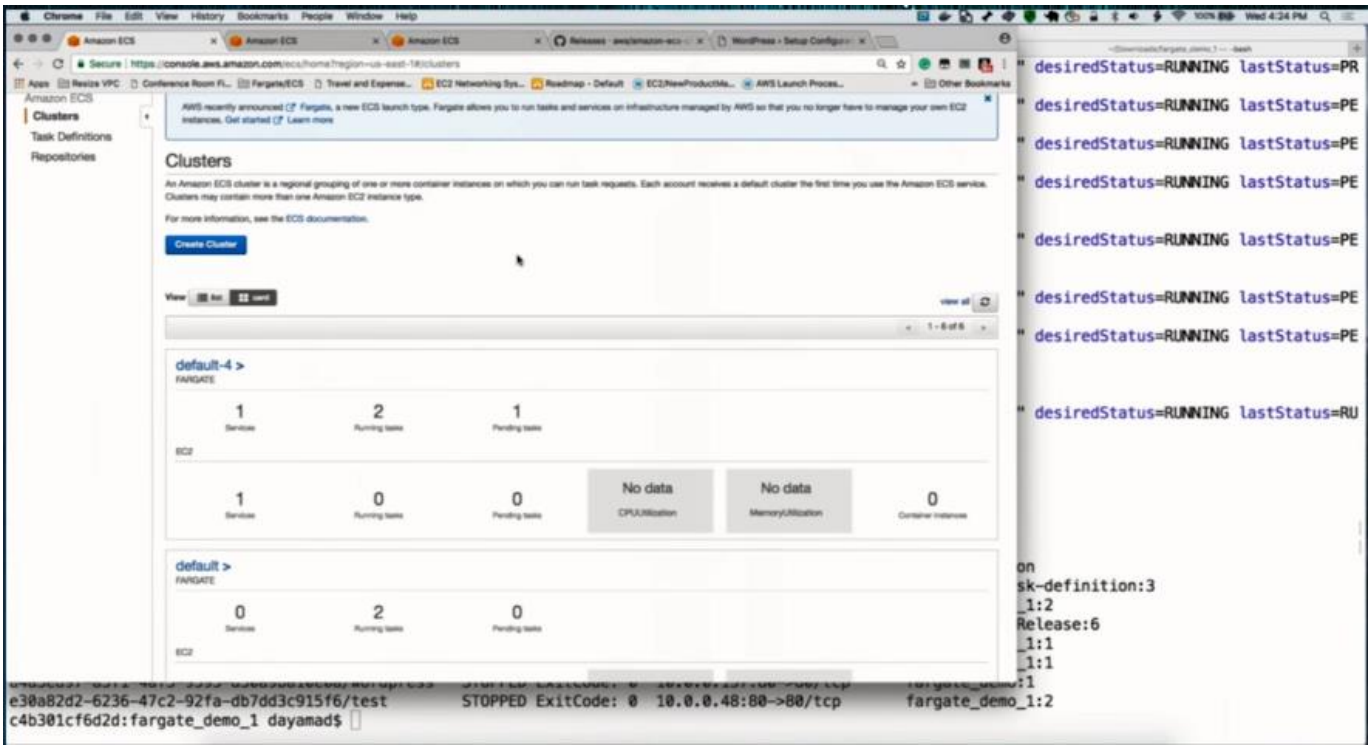
```
c4b301cf6d2d:fargate_demo_1 dayamad$ ecs-cli compose up
WARN[0000] Skipping unsupported YAML option...           option name=networks
WARN[0000] Skipping unsupported YAML option for service...  option name=networks service name=test
INFO[0000] Using ECS task definition                     TaskDefinition="fargate_demo_1:2"
INFO[0002] Starting container...                         container="2f10bddd-75ad-48ce-94ce-88042f77b362/test"
INFO[0002] Describe ECS container status                 container="2f10bddd-75ad-48ce-94ce-88042f77b362/test" desiredStatus=RUNNING lastStatus=PR
OVISIONING taskDefinition="fargate_demo_1:2"
INFO[0014] Describe ECS container status                 container="2f10bddd-75ad-48ce-94ce-88042f77b362/test" desiredStatus=RUNNING lastStatus=PR
OVISIONING taskDefinition="fargate_demo_1:2"
INFO[0026] Describe ECS container status                 container="2f10bddd-75ad-48ce-94ce-88042f77b362/test" desiredStatus=RUNNING lastStatus=PR
OVISIONING taskDefinition="fargate_demo_1:2"
INFO[0039] Describe ECS container status                 container="2f10bddd-75ad-48ce-94ce-88042f77b362/test" desiredStatus=RUNNING lastStatus=PE
NDING taskDefinition="fargate_demo_1:2"
INFO[0051] Describe ECS container status                 container="2f10bddd-75ad-48ce-94ce-88042f77b362/test" desiredStatus=RUNNING lastStatus=PE
NDING taskDefinition="fargate_demo_1:2"
INFO[0063] Describe ECS container status                 container="2f10bddd-75ad-48ce-94ce-88042f77b362/test" desiredStatus=RUNNING lastStatus=PE
NDING taskDefinition="fargate_demo_1:2"

INFO[0075] Describe ECS container status                 container="2f10bddd-75ad-48ce-94ce-88042f77b362/test" desiredStatus=RUNNING lastStatus=PE
NDING taskDefinition="fargate_demo_1:2"

INFO[0088] Describe ECS container status                 container="2f10bddd-75ad-48ce-94ce-88042f77b362/test" desiredStatus=RUNNING lastStatus=PE
NDING taskDefinition="fargate_demo_1:2"
INFO[0100] Describe ECS container status                 container="2f10bddd-75ad-48ce-94ce-88042f77b362/test" desiredStatus=RUNNING lastStatus=PE
NDING taskDefinition="fargate_demo_1:2"


INFO[0112] Started container...                          container="2f10bddd-75ad-48ce-94ce-88042f77b362/test" desiredStatus=RUNNING lastStatus=RU
NNING taskDefinition="fargate_demo_1:2"
c4b301cf6d2d:fargate_demo_1 dayamad$
c4b301cf6d2d:fargate_demo_1 dayamad$
c4b301cf6d2d:fargate_demo_1 dayamad$
c4b301cf6d2d:fargate_demo_1 dayamad$
c4b301cf6d2d:fargate_demo_1 dayamad$
c4b301cf6d2d:fargate_demo_1 dayamad$ d
```



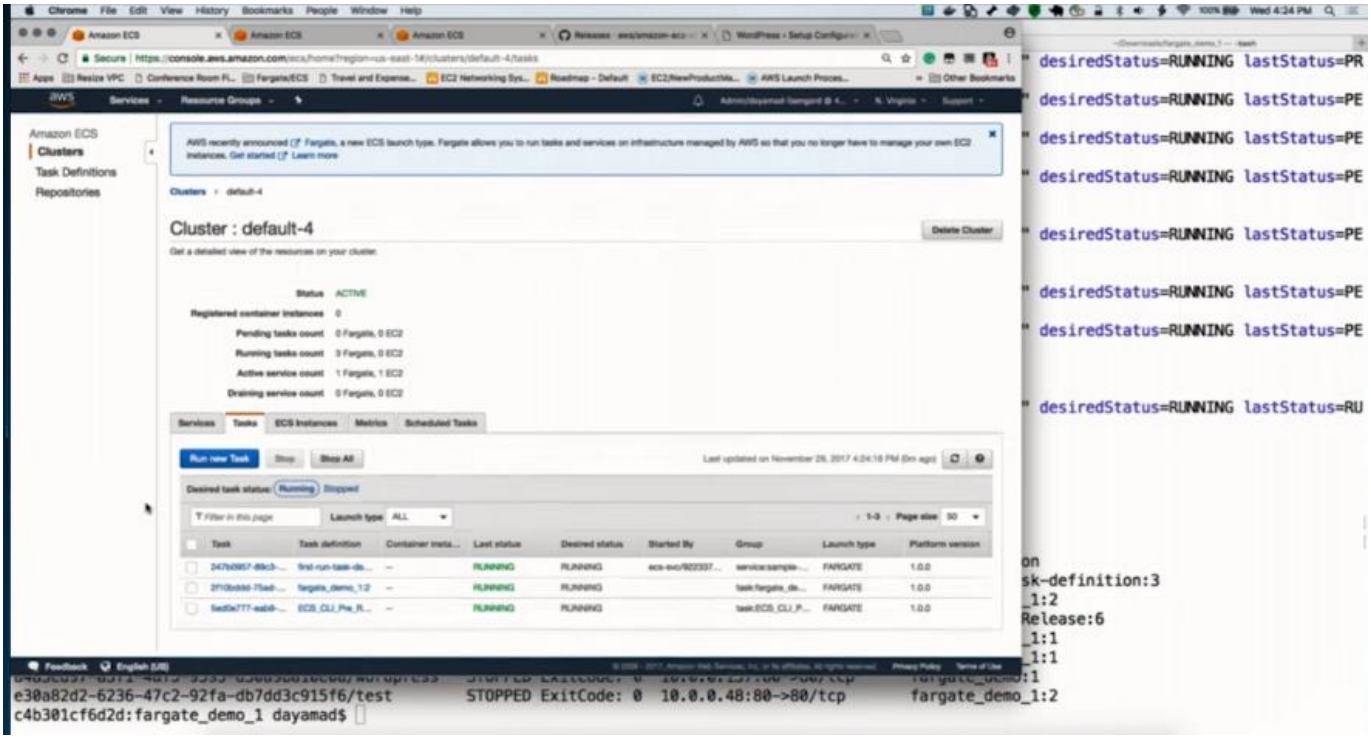
```
INFO[0026] Describe ECS container status                 container="2f10bddd-75ad-48ce-94ce-88042f77b362/test" desiredStatus=RUNNING lastStatus=PR
OVISIONING taskDefinition="fargate_demo_1:2"
INFO[0039] Describe ECS container status                 container="2f10bddd-75ad-48ce-94ce-88042f77b362/test" desiredStatus=RUNNING lastStatus=PE
NDING taskDefinition="fargate_demo_1:2"
INFO[0051] Describe ECS container status                 container="2f10bddd-75ad-48ce-94ce-88042f77b362/test" desiredStatus=RUNNING lastStatus=PE
NDING taskDefinition="fargate_demo_1:2"
INFO[0063] Describe ECS container status                 container="2f10bddd-75ad-48ce-94ce-88042f77b362/test" desiredStatus=RUNNING lastStatus=PE
NDING taskDefinition="fargate_demo_1:2"

INFO[0075] Describe ECS container status                 container="2f10bddd-75ad-48ce-94ce-88042f77b362/test" desiredStatus=RUNNING lastStatus=PE
NDING taskDefinition="fargate_demo_1:2"

INFO[0088] Describe ECS container status                 container="2f10bddd-75ad-48ce-94ce-88042f77b362/test" desiredStatus=RUNNING lastStatus=PE
NDING taskDefinition="fargate_demo_1:2"
INFO[0100] Describe ECS container status                 container="2f10bddd-75ad-48ce-94ce-88042f77b362/test" desiredStatus=RUNNING lastStatus=PE
NDING taskDefinition="fargate_demo_1:2"


INFO[0112] Started container...                          container="2f10bddd-75ad-48ce-94ce-88042f77b362/test" desiredStatus=RUNNING lastStatus=RU
NNING taskDefinition="fargate_demo_1:2"
c4b301cf6d2d:fargate_demo_1 dayamad$
c4b301cf6d2d:fargate_demo_1 dayamad$
c4b301cf6d2d:fargate_demo_1 dayamad$
c4b301cf6d2d:fargate_demo_1 dayamad$
c4b301cf6d2d:fargate_demo_1 dayamad$
c4b301cf6d2d:fargate_demo_1 dayamad$ ecs-cli ps
Name                                                State               Ports                     TaskDefinition
247b0957-89c3-4505-8b6c-796ddf164768/sample-app     RUNNING             34.231.147.31:80->80/tcp  first-run-task-definition:3
2f10bddd-75ad-48ce-94ce-88042f77b362/test           RUNNING             34.238.191.113:80->80/tcp  fargate_demo_1:2
5ed0e777-eab9-4196-8ca3-fef0d9cec5ca/wordpress      RUNNING             34.204.185.212:80->80/tcp  ECS_CLI_Pre_Release:6
26ef9994-463d-45aa-bd0a-f3187f6577fd/wordpress      STOPPED ExitCode: 0  10.0.0.176:80->80/tcp     fargate_demo_1:1
3220d2b0-5b29-499e-b313-1954634e65f5/wordpress      STOPPED ExitCode: 0  10.0.0.5:80->80/tcp       fargate_demo_1:1
a4a3ed97-a3f1-4af5-9595-d30a9ba10e0a/wordpress      STOPPED ExitCode: 0  10.0.0.157:80->80/tcp     fargate_demo:1
e30a82d2-6236-47c2-92fa-db7dd3c915f6/test           STOPPED ExitCode: 0  10.0.0.48:80->80/tcp      fargate_demo_1:2
c4b301cf6d2d:fargate_demo_1 dayamad$
```

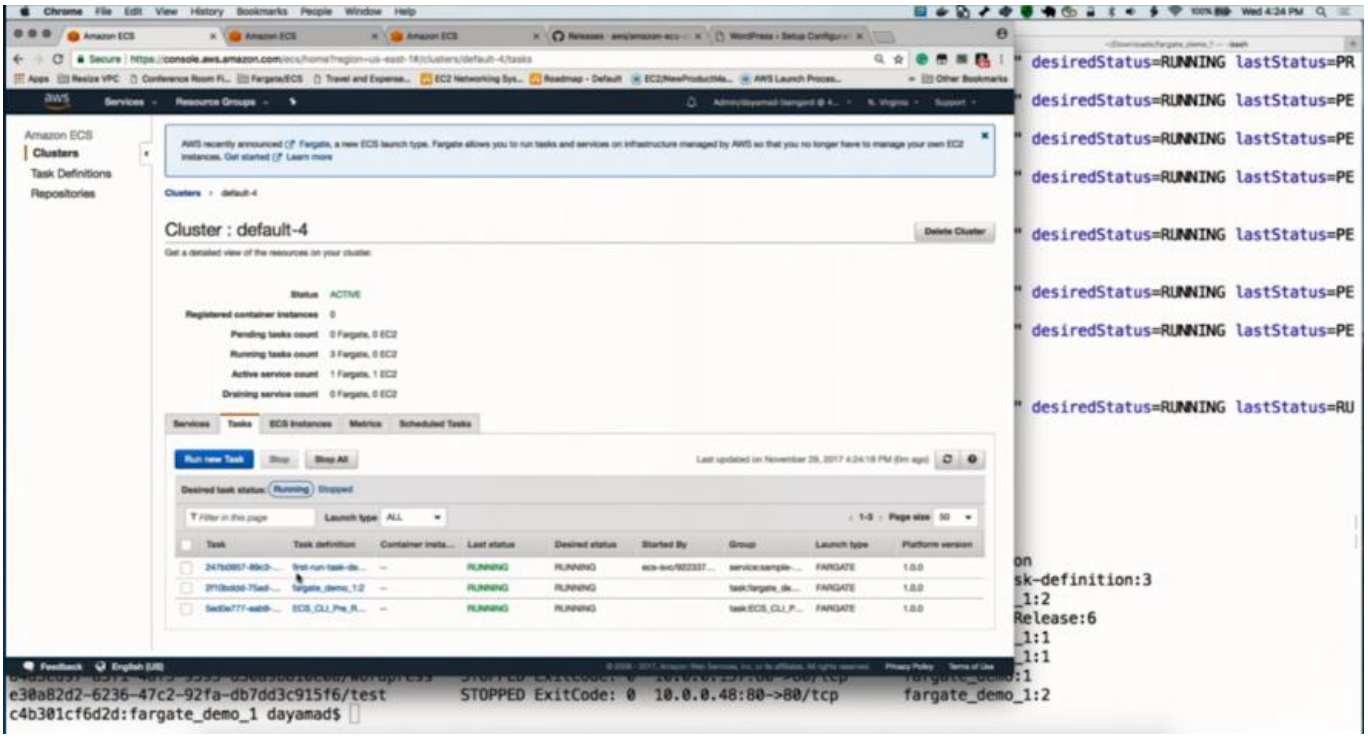


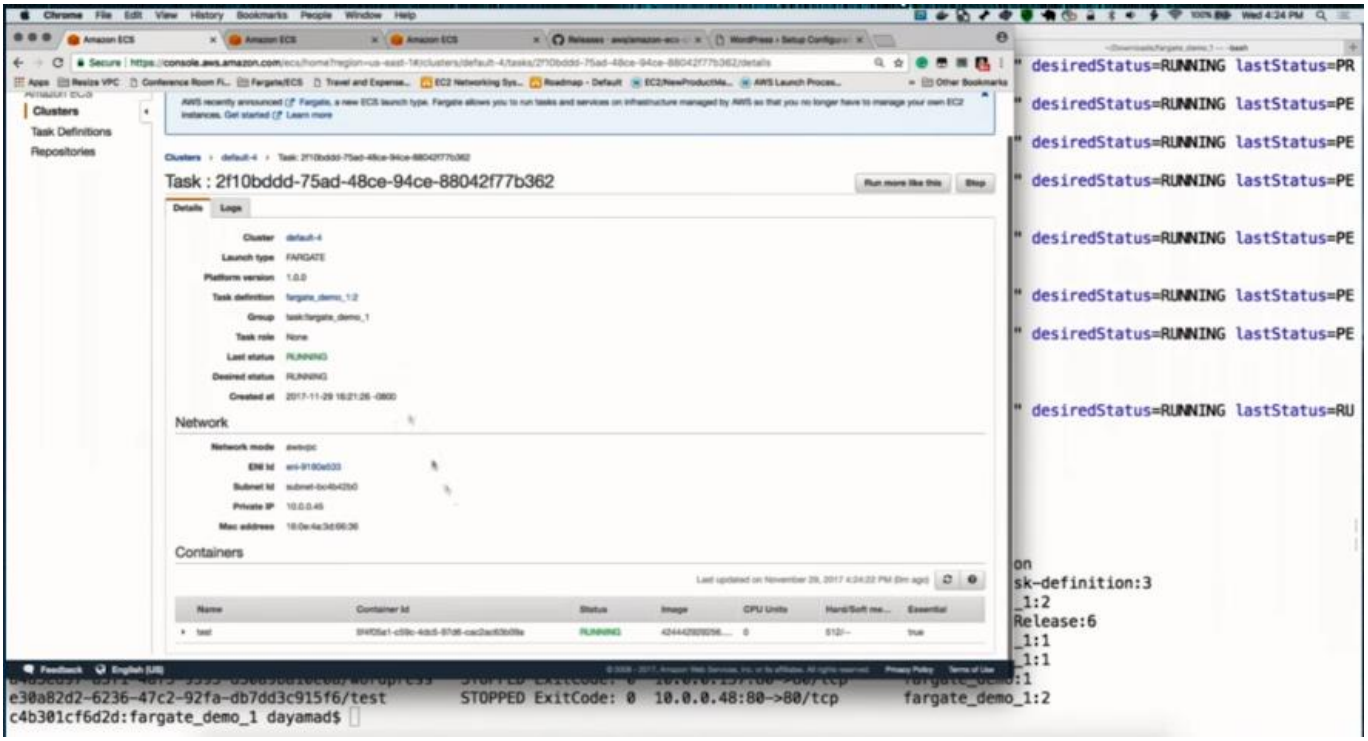
```
version: 1
task_definition:
  task_execution_role: arn:aws:iam::424442929256:role/ecsTaskExecutionRole
  ecs_network_mode: awsvpc
  task_size:
    mem_limit: 0.5GB
    cpu_limit: 256
run_params:
  network_configuration:
    awsvpc_configuration:
      subnets:
        - subnet-bc4b42b0
      assign_public_ip: ENABLED
      security_groups:
        - sg-341bb141
```



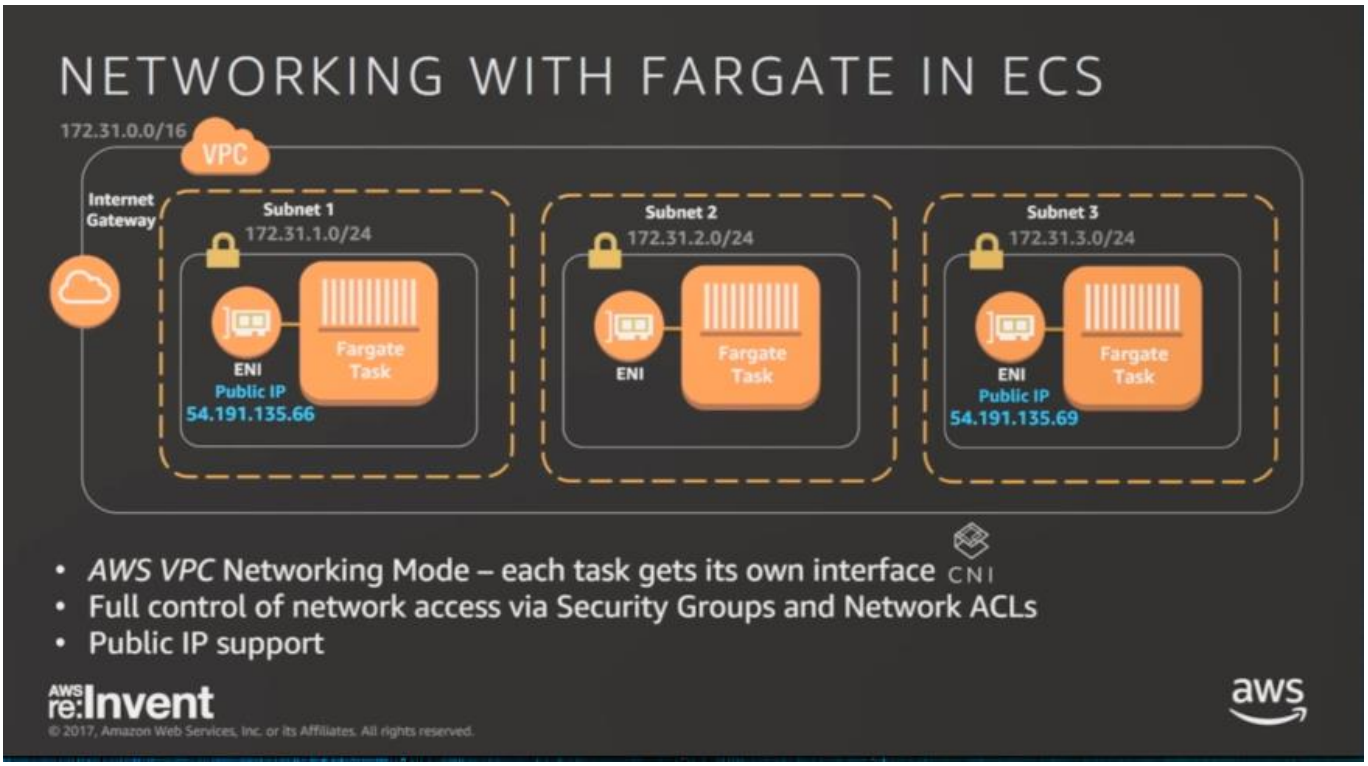
We have the Wordpress app running



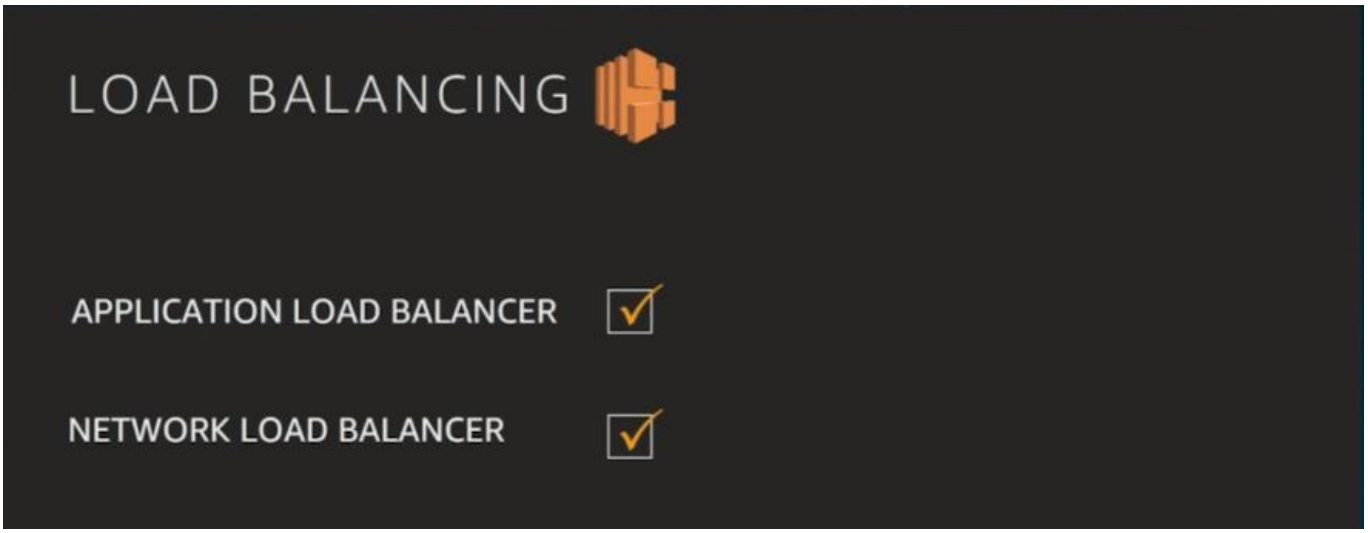




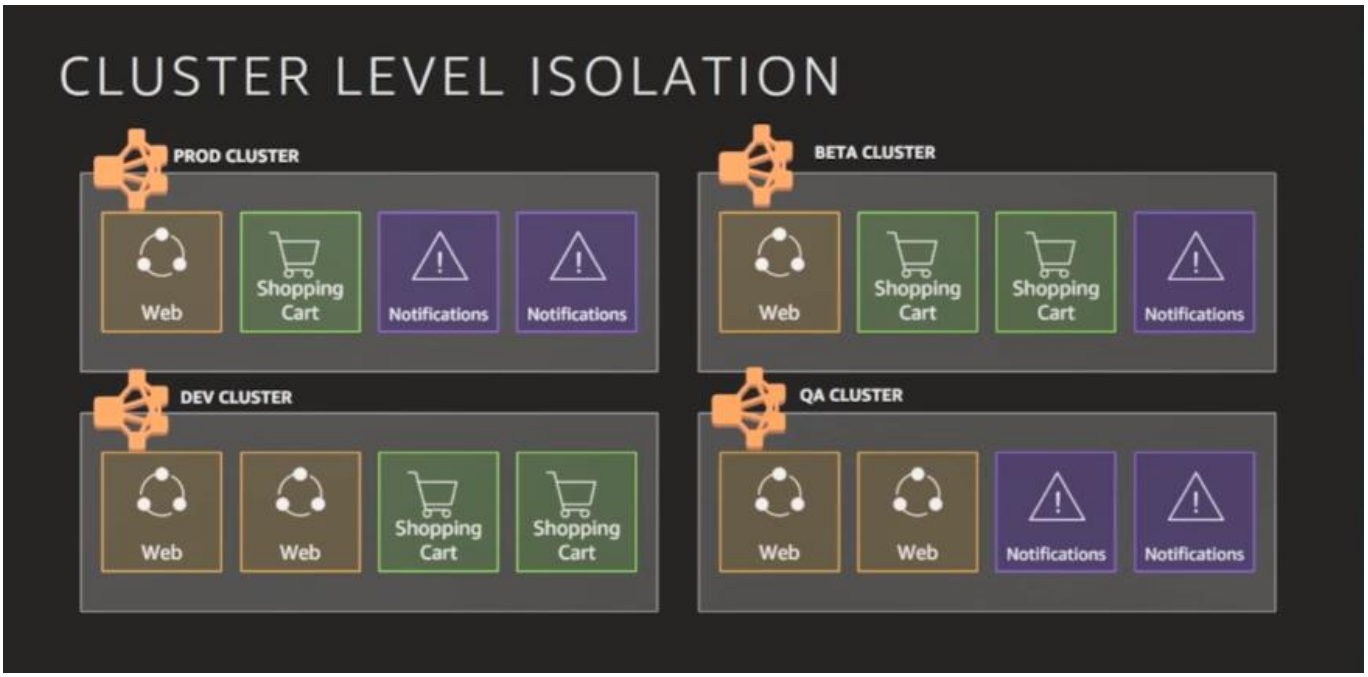
We have a dedicated ENI for the running task, nothing else to manage



In the Fargate model, even though there are no instances to manage, each task gets its ENI in a separate networked namespace, that namespace is shared by all the containers within the same task. This gives you complete control over the security policies for the tasks regarding security groups, NACLs, routing, etc. You can designate a subnet for Fargate to place your app in with the specified ENI.

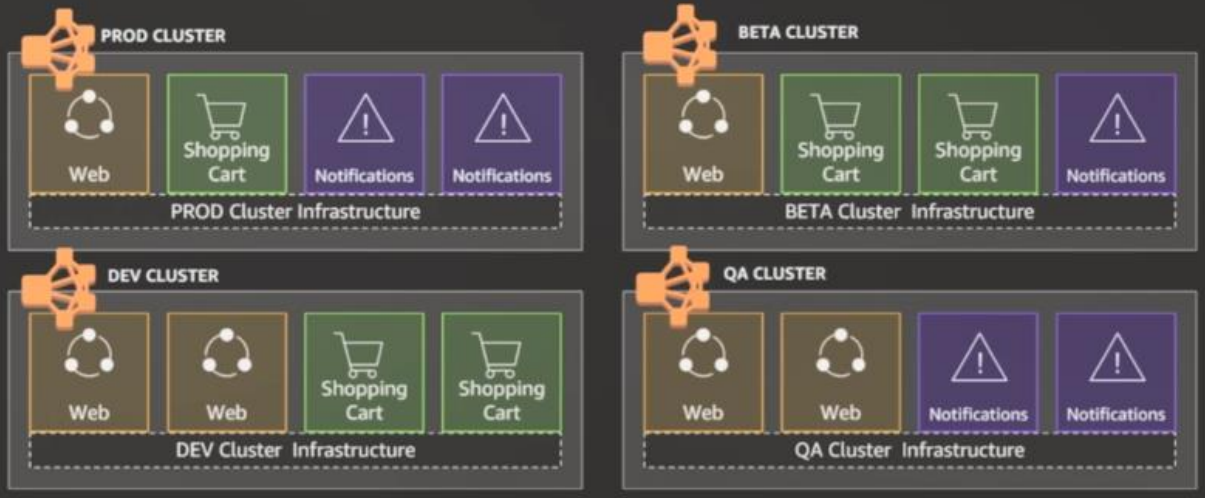


Let us talk about how you can implicate security with your clusters using Fargate.

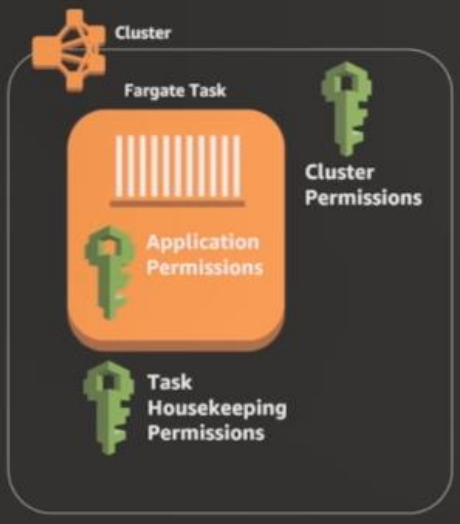


Fargate still uses the notion of the cluster even though the cluster does not really have any registered instances anymore.

CLUSTER LEVEL ISOLATION



PERMISSION TIERS



Cluster Permissions:

Who can run/see tasks in the cluster?

Application (Task) Permissions:

Which of my AWS resources can this application access?

Housekeeping Permissions:

What permissions do I want to grant ECS to perform?

e.g.

- ECR Image Pull
- CloudWatch Logs pushing
- ENI creation
- Register/Deregister targets into ELB

CONTAINER REGISTRIES

REGISTRY SUPPORT

Amazon Elastic Container Registry (ECR)

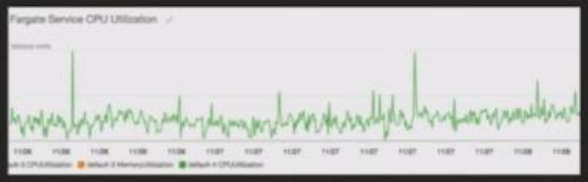
Public Repositories supported

3rd Party Private Repositories (coming soon!)

VISIBILITY AND MONITORING

CloudWatch Logs
CloudWatch Events supported

Service-level metrics available

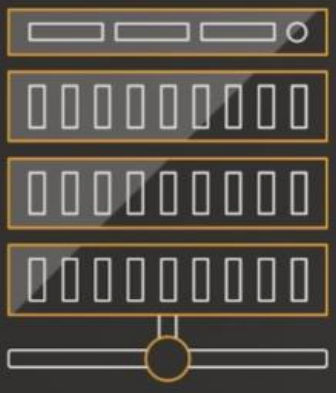


STORAGE

Ephemeral storage backed by EBS

Container Storage Space – 10GB

Shared volume space for containers within the task – 4GB



CONFIGURATIONS & PRICING

PRICING DIMENSIONS

Dimensions: Task level CPU and memory

Per-second billing

Task level resources

- Configurable independently (within a range)

```
{
"memory": "1 vCPU",
"cpu": "3GB",
"networkMode": "AWSVPC",
"compatibilities": ["FARGATE",
"EC2"],
"placementConstraints": [],
  "containerDefinitions": [
    {

<snip>…....
```

TASK CPU & MEMORY CONFIGURATIONS

Flexible configuration options –
50 CPU/memory configurations

CPU	Memory
256 (.25 vCPU)	512MB, 1GB, 2GB
512 (.5 vCPU)	1GB, 2GB, 3GB, 4GB
1024 (1 vCPU)	2GB, 3GB, 4GB, 5GB, 6GB, 7GB, 8GB
2048 (2 vCPU)	Between 4GB and 16GB in 1GB increments
4096 (4 vCPU)	Between 8GB and 30GB in 1GB increments

CLOUDFORMATION SUPPORT


99.99%


Amazon ECS CLUSTER
Subnet 1
172.31.1.0/24
Subnet 2
172.31.2.0/24
Subnet 3
172.31.3.0/24
Web
Shopping Cart
Notifications
Availability Zone #1
Availability Zone #2
Availability Zone #3
AWS re:Invent
© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.
aws

Note that the Fargate and EC2 stacks are running within the same VPC and AZs and can communicate with each other as if they are not different

PARTNER INTEGRATIONS

OTHER RELEVANT SESSIONS

CON333 – Deep Dive into AWS Fargate

CON201 – Containers on AWS – State of the Union

CON404 – Deep Dive into Container Scheduling with Amazon ECS

CON401 – Container Networking Deep Dive with Amazon ECS

CON402 – Advanced Patterns in Microservices Implementation with Amazon ECS

THANK YOU

https://aws.amazon.com/fargate