

CON214

# INTRODUCTION TO AWS FARGATE

Anthony Suarez – GM, Amazon ECS & ECR

Deepak Dayama – Senior Product Manager, Amazon ECS

November 29, 2017

**AWS**  
**re:Invent**

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



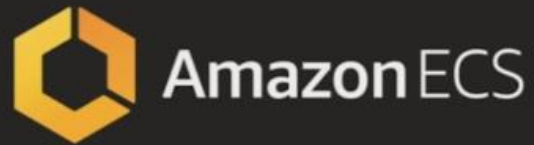
## OUR JOURNEY!

**AWS**  
**re:Invent**

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



# DAY ONE!

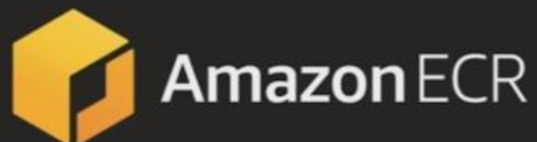
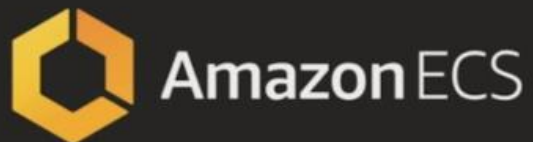


**AWS**  
**re:Invent**

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



# BUILDING AN ECOSYSTEM



**AWS**  
**re:Invent**

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



# WHY DO WE LOVE CONTAINERS?



Packaging



Distribution



Immutable  
infrastructure

AWS  
re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



# HELPING CUSTOMERS SCALE CONTAINERS



**450+%**  
growth



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

AWS  
re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



# PRODUCTION WORKLOADS ON AWS



AWS VPC  
networking mode



Advanced task  
placement



Deep integration  
with AWS services



ECS CLI



Global footprint



Powerful scheduling  
engines



Auto scaling



CloudWatch metrics



Load balancers

AWS  
re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



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

# CUSTOMERS ARE OUR KEY!



**50+**  
releases  
since 2015

AWS  
re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



# ENABLE FOCUS ON APPLICATIONS



AWS  
re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



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.

# INTRODUCING FARGATE!



# Fargate

AWS  
re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



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.

# CHANGING COMPUTE CONSUMPTION MODEL



No instances  
to manage



Task  
native API



Resource  
based pricing



Simple, easy to use,  
powerful – and new  
consumption model

AWS  
re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



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

# PRODUCTION WORKLOADS ON AWS



Fargate



AWS VPC  
networking mode



Advanced task  
placement



Deep integration  
with AWS services



ECS CLI



Global footprint (in 2018)



Powerful scheduling  
engines



Task auto scaling



CloudWatch metrics



Load balancers

AWS  
re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



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



# EKS SUPPORT FOR FARGATE IN 2018



Fargate



AmazonEKS

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

## FARGATE: UNDER THE HOOD



Fargate

## MICROSERVICES



## BATCH JOBS



## MIGRATION TO THE CLOUD





# HOW DO I RUN CONTAINERS ON FARGATE?

AWS  
re:Invent



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

## RUNNING CONTAINER



AWS  
re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



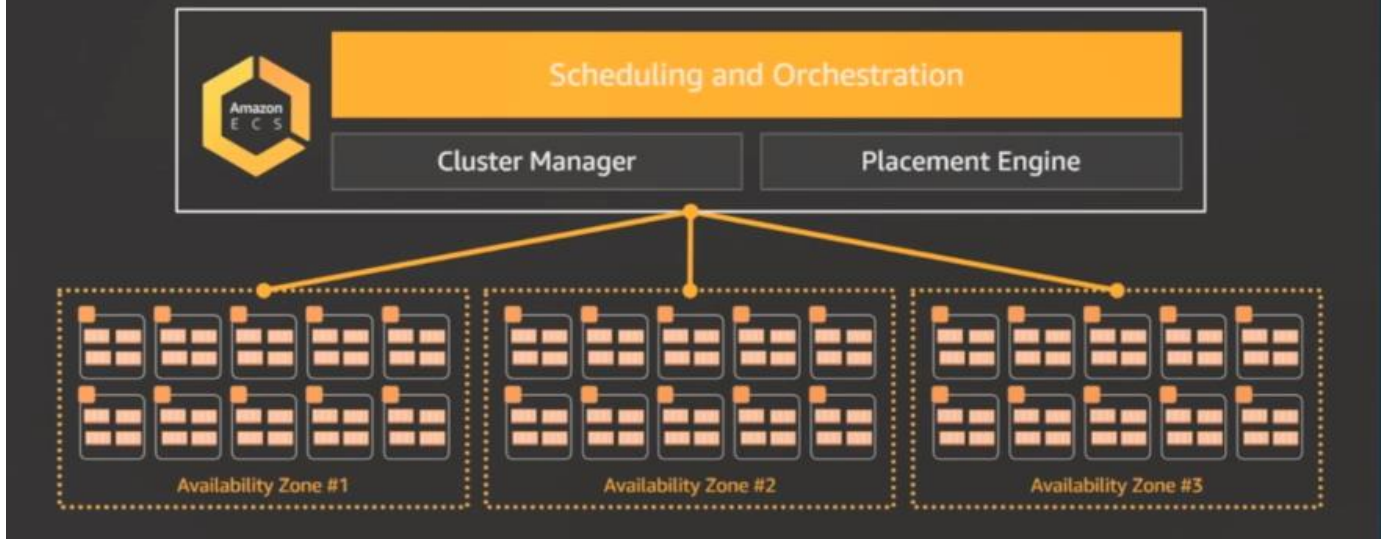
# RUNNING CONTAINERS



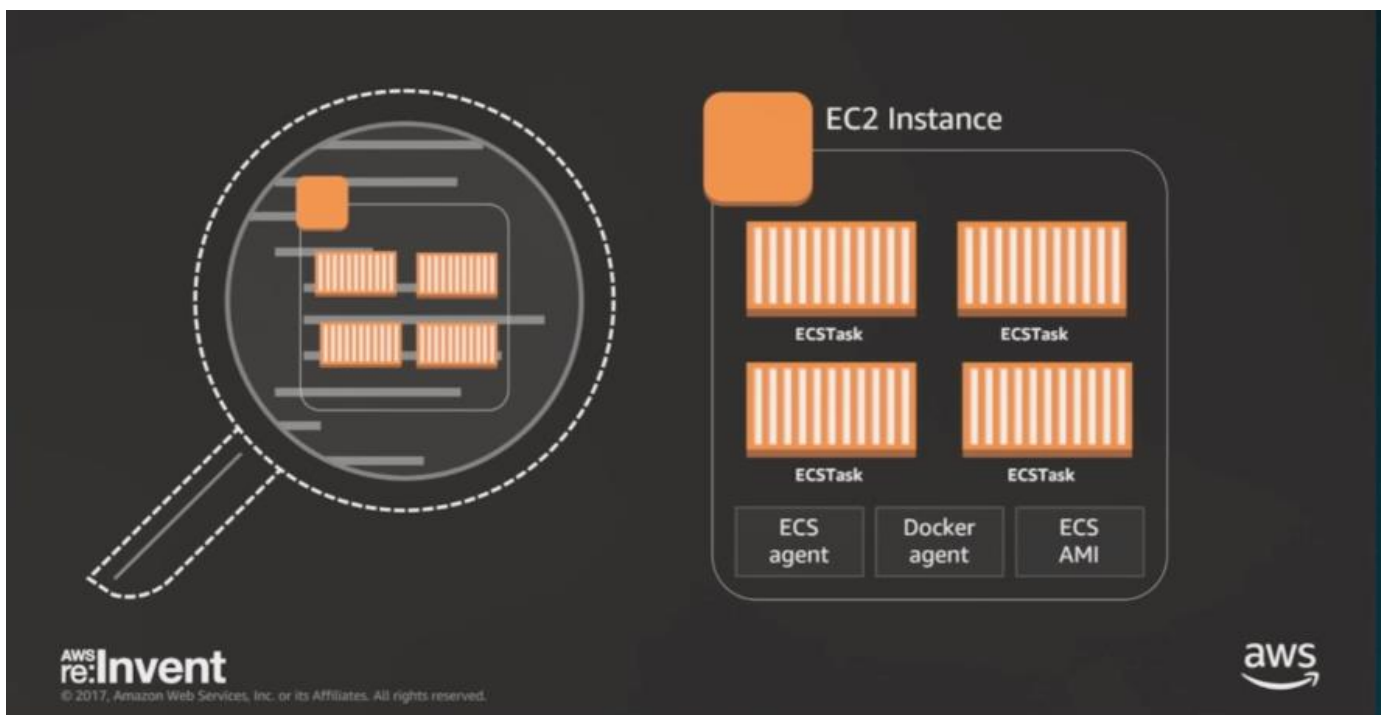
# RUNNING CONTAINERS AT SCALE WITH ECS



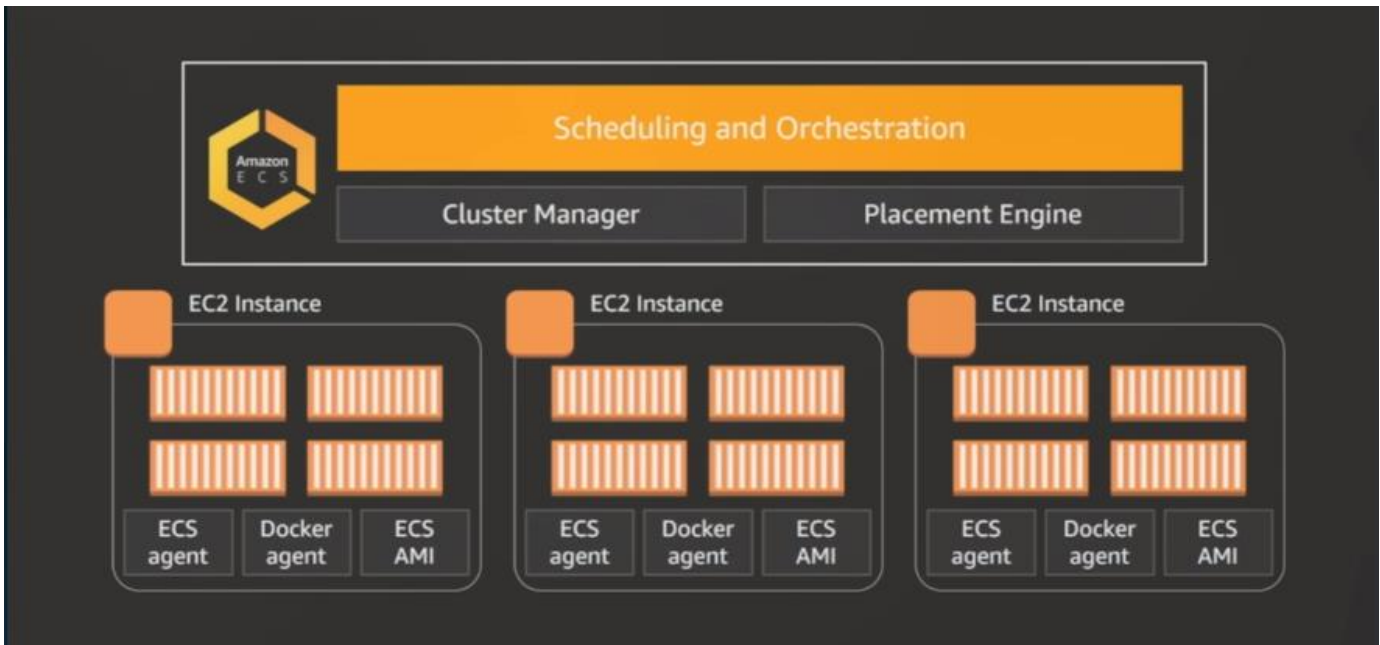
# RUNNING CONTAINERS AT SCALE WITH ECS



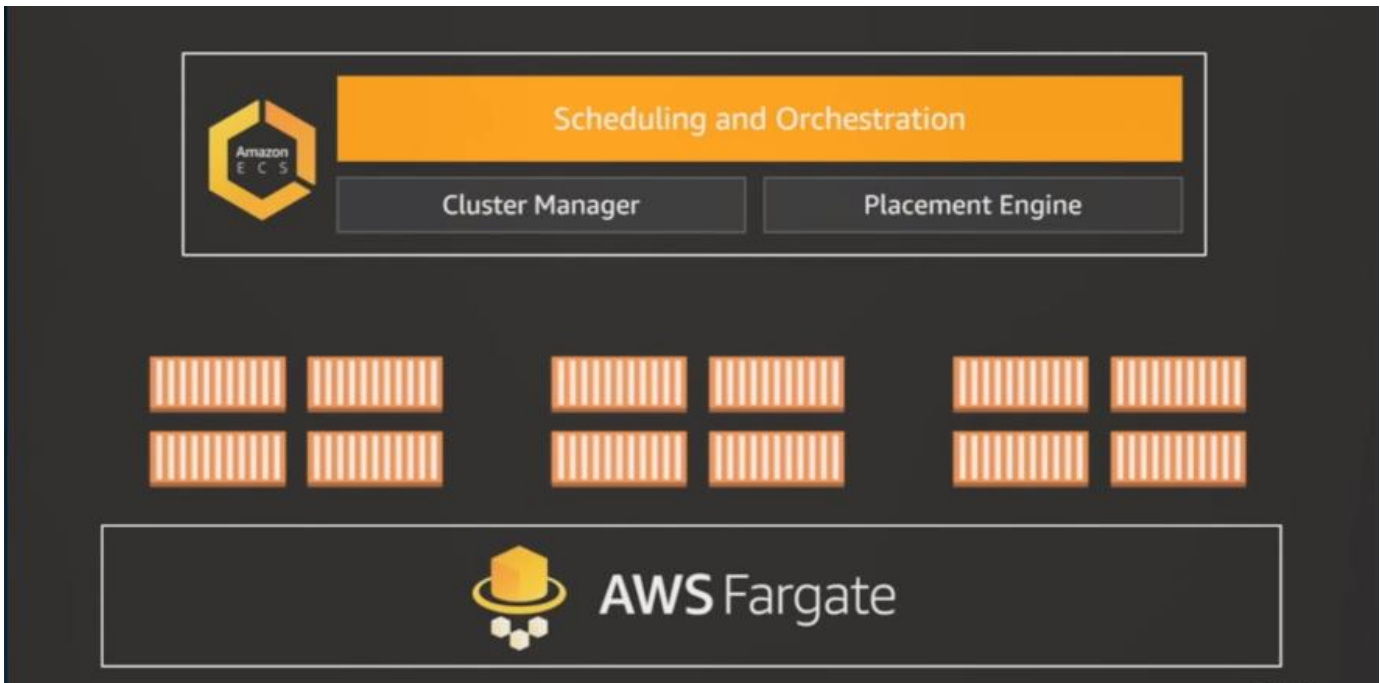
ECS allows you to be able to operationalize the scale using the cluster manage and placement engine



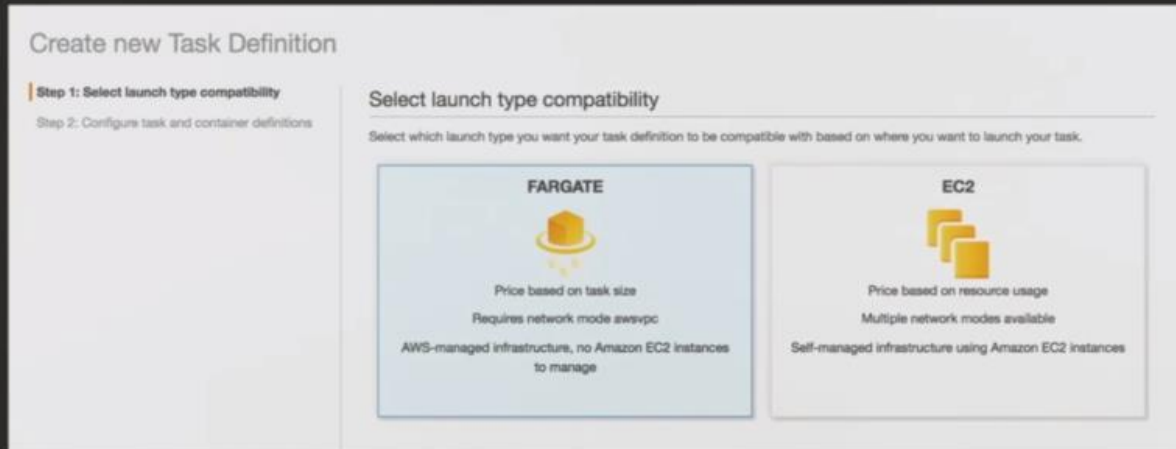
You also have to manage the Docker daemon and other things as above



Fargate enables you to stop managing things above as below



# RUNNING FARGATE CONTAINERS WITH ECS



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

# RUNNING FARGATE CONTAINERS WITH ECS



*Same Task Definition* schema



Use ECS APIs to launch Fargate Containers



Easy migration – Run *Fargate* and *EC2* launch type tasks in the same cluster

## DEMO



```
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-88042f77b362/test"
INFO[0002] Describe ECS container status container="2f10bddd-75ad-48ce-88042f77b362/test" desiredStatus=RUNNING lastStatus=PROVISIONING 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
~
~
```

The screenshot shows the Amazon ECS console for a cluster named 'default-4'. The cluster status is 'ACTIVE'. Under 'Registered container instances', there are 3 instances. The 'Task Definition' section shows a table of tasks:

Task	Task definition	Container instance	Last status	Desired status	Started by	Group	Launch type	Platform version
ecs-t111-ec2-1	ecs-t111-ec2-1	ami-ec2-111-ec2-1	RUNNING	RUNNING	aws-ec2-111-ec2-1	ecs-t111-ec2-1	EC2	1.0.0
ecs-t111-ec2-2	ecs-t111-ec2-2	ami-ec2-111-ec2-2	RUNNING	RUNNING	aws-ec2-111-ec2-2	ecs-t111-ec2-2	EC2	1.0.0

The screenshot shows the Amazon ECS console's 'Clusters' page. At the top, there's a navigation bar with 'Clusters' selected. Below it, a brief description of ECS clusters is provided. A 'Create Cluster' button is visible. The main content area shows a table of ECS clusters. The first cluster, 'default-4', is of type 'FARGATE' and has 1 service, 2 running tasks, and 1 pending task. The second cluster, 'default', is also of type 'FARGATE' and has 0 services, 2 running tasks, and 0 pending tasks. The table also shows metrics for CPU utilization, memory utilization, and container instances.

Cluster Name	Type	Services	Running Tasks	Pending Tasks	CPU Utilization	Memory Utilization	Container Instances
default-4	FARGATE	1	2	1	No data	No data	0
default	FARGATE	0	2	0	No data	No data	0



```
Terminal Shell Edit View Window Help
fargate_demo_1 -- bash -- 148x33
c4b301cf6d2d:fargate_demo_1 dayamad$ ecs-cli compose up
WARN[0000] Skipping unsupported YAML option...
WARN[0000] Skipping unsupported YAML option for service...
INFO[0000] Using ECS task definition
INFO[0002] Starting container...
INFO[0002] Describe ECS container status
OVISIONING taskDefinition="fargate_demo_1:2"
INFO[0014] Describe ECS container status
OVISIONING taskDefinition="fargate_demo_1:2"
INFO[0026] Describe ECS container status
OVISIONING taskDefinition="fargate_demo_1:2"
INFO[0039] Describe ECS container status
NDING taskDefinition="fargate_demo_1:2"
INFO[0051] Describe ECS container status
NDING taskDefinition="fargate_demo_1:2"
INFO[0063] Describe ECS container status
NDING taskDefinition="fargate_demo_1:2"

INFO[0075] Describe ECS container status
NDING taskDefinition="fargate_demo_1:2"

INFO[0088] Describe ECS container status
NDING taskDefinition="fargate_demo_1:2"
INFO[0100] Describe ECS container status
NDING taskDefinition="fargate_demo_1:2"

INFO[0112] Started container...
NDING 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$
c4b301cf6d2d:fargate_demo_1 dayamad$ d

option name=networks
TaskDefinition="fargate_demo_1:2"
container="2f10bddd-75ad-48ce-94ce-88042f77b362/test" desiredStatus=RUNNING lastStatus=PR
container="2f10bddd-75ad-48ce-94ce-88042f77b362/test" desiredStatus=RUNNING lastStatus=PR
container="2f10bddd-75ad-48ce-94ce-88042f77b362/test" desiredStatus=RUNNING lastStatus=PR
container="2f10bddd-75ad-48ce-94ce-88042f77b362/test" desiredStatus=RUNNING lastStatus=PE
container="2f10bddd-75ad-48ce-94ce-88042f77b362/test" desiredStatus=RUNNING lastStatus=PE
container="2f10bddd-75ad-48ce-94ce-88042f77b362/test" desiredStatus=RUNNING lastStatus=PE
container="2f10bddd-75ad-48ce-94ce-88042f77b362/test" desiredStatus=RUNNING lastStatus=PE
container="2f10bddd-75ad-48ce-94ce-88042f77b362/test" desiredStatus=RUNNING lastStatus=PE
container="2f10bddd-75ad-48ce-94ce-88042f77b362/test" desiredStatus=RUNNING lastStatus=RU
```

```
Terminal Shell Edit View Window Help
fargate_demo_1 -- bash -- 148x33
INFO[0026] Describe ECS container status
OVISIONING taskDefinition="fargate_demo_1:2"
INFO[0039] Describe ECS container status
NDING taskDefinition="fargate_demo_1:2"
INFO[0051] Describe ECS container status
NDING taskDefinition="fargate_demo_1:2"
INFO[0063] Describe ECS container status
NDING taskDefinition="fargate_demo_1:2"

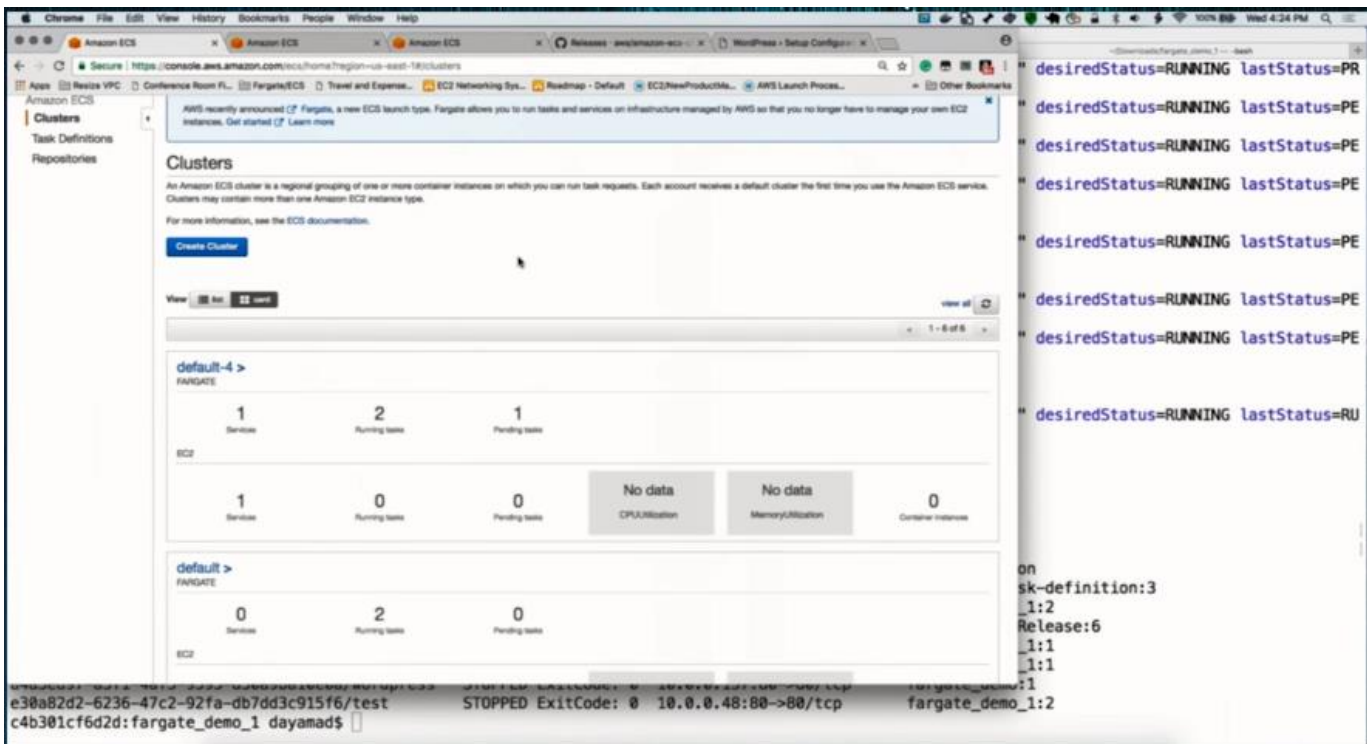
INFO[0075] Describe ECS container status
NDING taskDefinition="fargate_demo_1:2"

INFO[0088] Describe ECS container status
NDING taskDefinition="fargate_demo_1:2"
INFO[0100] Describe ECS container status
NDING taskDefinition="fargate_demo_1:2"

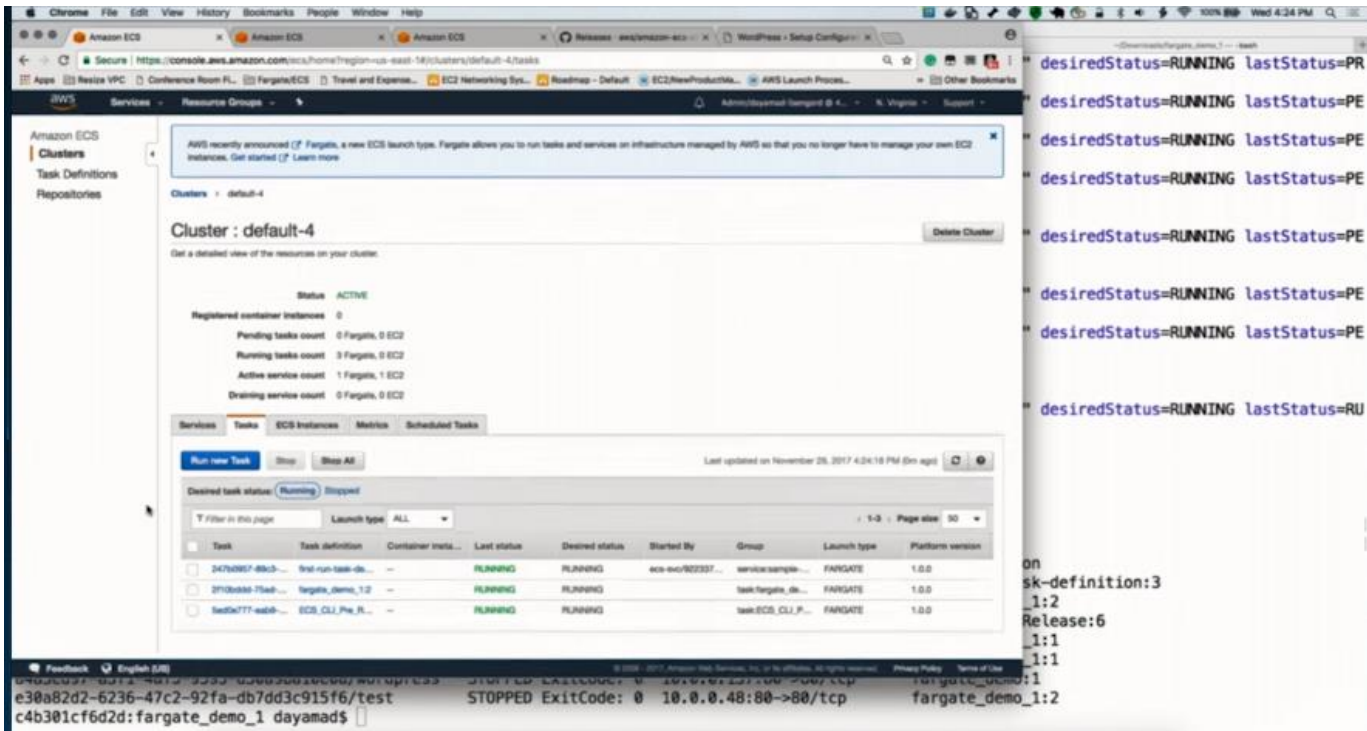
INFO[0112] Started container...
NDING taskDefinition="fargate_demo_1:2"
c4b301cf6d2d:fargate_demo_1 dayamad$
c4b301cf6d2d:fargate_demo_1 dayamad$
c4b301cf6d2d:fargate_demo_1 dayamad$
c4b301cf6d2d:fargate_demo_1 dayamad$
c4b301cf6d2d:fargate_demo_1 dayamad$
c4b301cf6d2d:fargate_demo_1 dayamad$
c4b301cf6d2d:fargate_demo_1 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$
```

```
Terminal Shell Edit View Window Help
fargate_demo_1 -- ECS-Param -- vi ecs-param.yml -- 136x33
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





Amazon ECS console showing the 'default-4' cluster. The cluster is in an ACTIVE state. The 'Tasks' tab shows a list of tasks, including 'fargate\_demo\_1,2' which is in a RUNNING state. The 'Services' tab shows a list of services, including 'fargate\_demo\_1' which is in a RUNNING state. The 'Task Definition' tab shows the details for 'fargate\_demo\_1,2', including the task definition 'fargate\_demo\_1,2' and the task role 'None'.

```
desiredStatus=RUNNING lastStatus=PR
desiredStatus=RUNNING lastStatus=PE
desiredStatus=RUNNING lastStatus=PE
desiredStatus=RUNNING lastStatus=PE
desiredStatus=RUNNING lastStatus=PE
desiredStatus=RUNNING lastStatus=PE
desiredStatus=RUNNING lastStatus=RU
```

on  
sk-definition:3  
\_1:2  
Release:6  
\_1:1  
\_1:1  
\_1:1  
\_1:1

```
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$
```

Amazon ECS console showing the details for the task '2f10bdd-75ad-48ce-94ce-88042f77b362'. The task is in a RUNNING state. The 'Network' tab shows the network details, including the network mode 'awotpc', the ENI ID 'eni-918a6033', the subnet ID 'subnet-ba4b429d', the private IP '10.0.0.48', and the MAC address '18:00:0a:3d:00:30'. The 'Containers' tab shows a list of containers, including 'test' which is in a RUNNING state.

```
desiredStatus=RUNNING lastStatus=PR
desiredStatus=RUNNING lastStatus=PE
desiredStatus=RUNNING lastStatus=PE
desiredStatus=RUNNING lastStatus=PE
desiredStatus=RUNNING lastStatus=PE
desiredStatus=RUNNING lastStatus=PE
desiredStatus=RUNNING lastStatus=RU
```

on  
sk-definition:3  
\_1:2  
Release:6  
\_1:1  
\_1:1  
\_1:1  
\_1:1

```
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$
```

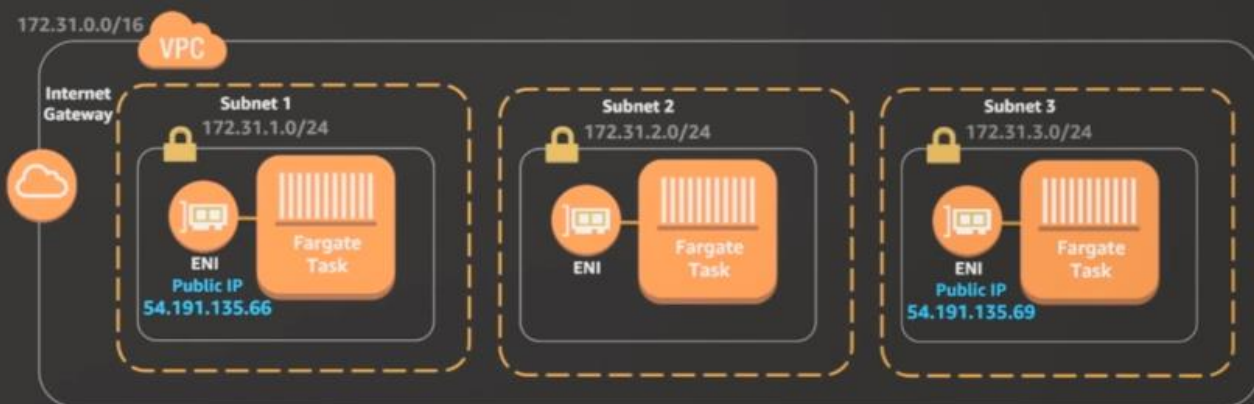
We have a dedicated ENI for the running task, nothing else to manage

DEMO

# NETWORKING



## NETWORKING WITH FARGATE IN ECS



- AWS VPC Networking Mode – each task gets its own interface CNI
- Full control of network access via Security Groups and Network ACLs
- Public IP support

AWS  
re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



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.

# LOAD BALANCING

APPLICATION LOAD BALANCER ☒

NETWORK LOAD BALANCER ☒

## SECURITY



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

## CLUSTER LEVEL ISOLATION



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

# CLUSTER LEVEL ISOLATION



AWS re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



## 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

AWS re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



# CONTAINER REGISTRIES



## REGISTRY SUPPORT

Amazon Elastic Container Registry (ECR)



---

Public Repositories supported



---

3<sup>rd</sup> Party Private Repositories (coming soon!)



**AWS**  
**re:Invent**

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



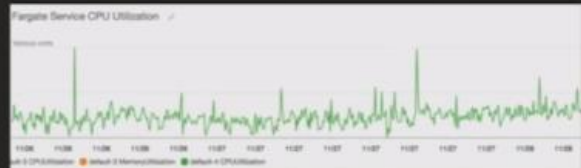


# VISIBILITY AND MONITORING

CloudWatch Logs  
CloudWatch Events supported



Service-level metrics available



**AWS re:Invent**

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

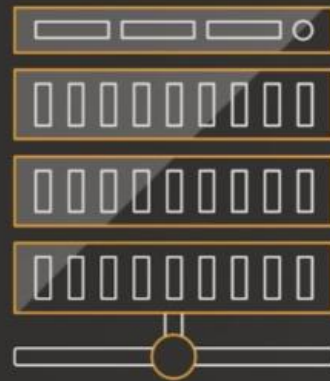


# STORAGE

Ephemeral storage backed by EBS

Container Storage Space – 10GB

Shared volume space for containers within the task – 4GB



**AWS re:Invent**

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



# 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>.....  
    }  
  ]  
}
```

**AWS**  
**re:Invent**

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



# 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

**AWS**  
**re:Invent**

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



## CLOUDFORMATION SUPPORT





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



DATADOG

splunk>

aqua

shippable



New Relic.



Twistlock™

AWS  
re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



## 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>