



CON314

Deploy a Deep Learning Framework on Amazon ECS

Chad Schmutz, Solutions Architect

Hubert Cheung, Solutions Architect

David Kuo, Solutions Architect

Andy Mui, Solutions Architect

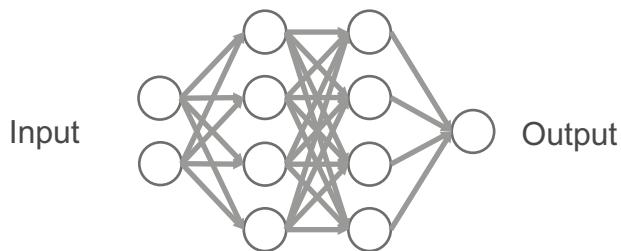
December 2016

What to expect from this workshop

- Introduce MXNet
- Containers
- Overview of ECS + ECR
- Overview of AWS CloudFormation
- Overview of EC2 Spot Fleet / Spot Instances
- Hands on Workshop

What's MXNet?

- Open-source deep learning framework - <https://github.com/dmlc/mxnet>
- Define, train, and deploy deep neural networks
- Highly scalable – single/multiple hosts, CPU/GPU support
- Support for multiple languages



dmlc
mxnet

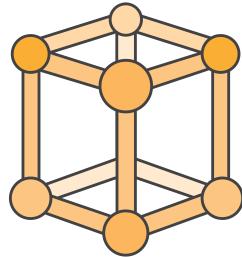


Why Containers?

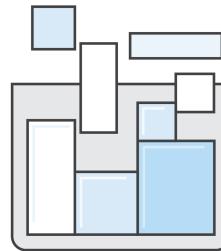
- Increase infrastructure utilization
- Environment isolation and fidelity
- Run diverse applications on shared hardware
- Changes are tracked
- Easy to deploy

ECS + ECR

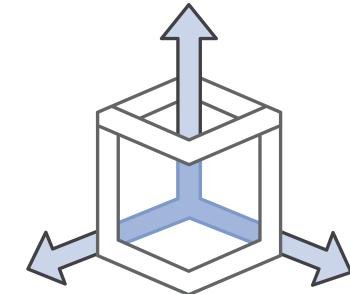
ECS Benefits



Cluster management
made easy



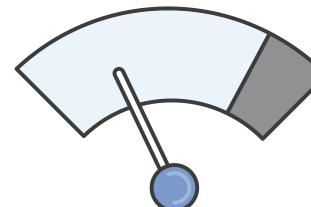
Flexible scheduling



Integrated and
extensible

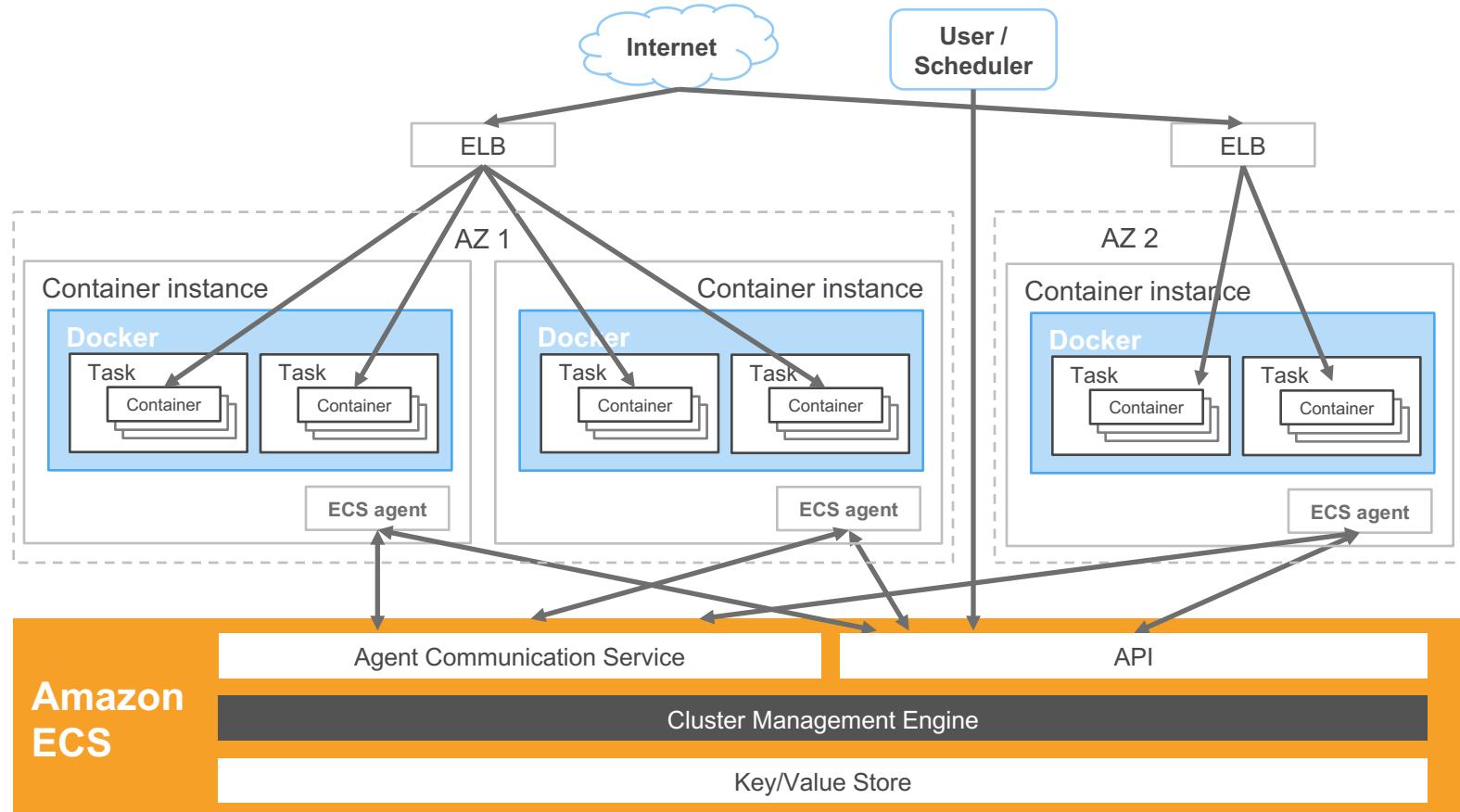


Security



Performance at scale

ECS Architecture



What's ECR?

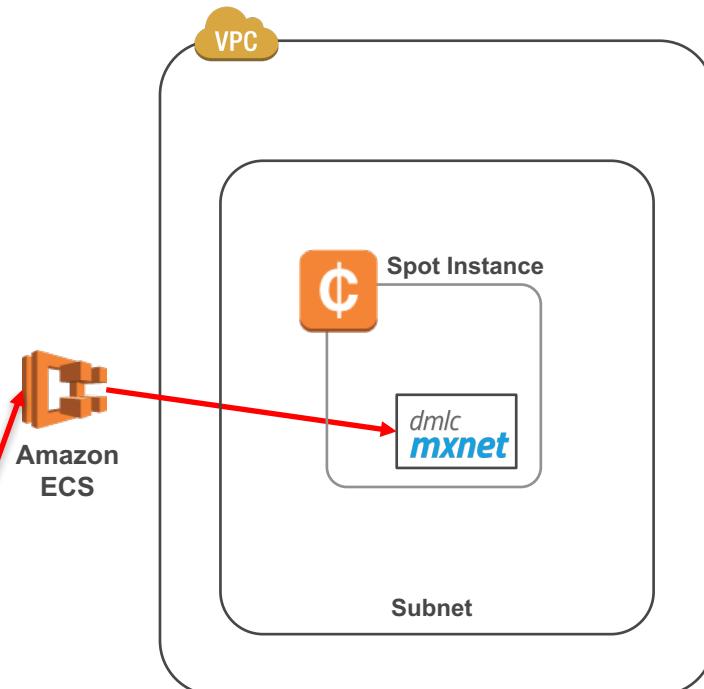
- Amazon EC2 Container Registry (ECR) is a fully-managed Docker container registry that makes it easy for developers to store, manage, and deploy Docker container images. Amazon ECR is integrated with Amazon EC2 Container Service (ECS), simplifying your development to production workflow.
- Learn More: <https://aws.amazon.com/ecr/>

How does ECS use ECR?



Amazon
ECR

```
"containerDefinitions": [
  {
    "volumesFrom": [],
    "memory": 2048,
    "extraHosts": null,
    "dnsServers": null,
    "disableNetworking": null,
    "dnsSearchDomains": null,
    "portMappings": [
      {
        "hostPort": 80,
        "containerPort": 8888,
        "protocol": "tcp"
      }
    ],
    "hostname": null,
    "essential": true,
    "entryPoint": null,
    "mountPoints": [],
    "name": "mxnet",
    "ulimits": null,
    "dockerSecurityOptions": null,
    "environment": [],
    "links": null,
    "workingDirectory": null,
    "readonlyRootFilesystem": null,
    "image": "██████████ dkr.ecr.us-west-2.amazonaws.com/test:latest"
  }
]
```



AWS CloudFormation

CloudFormation – Components & Technology

Template



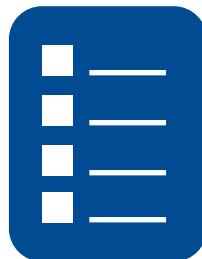
JSON formatted file

Parameter definition

Resource creation

Configuration actions

CloudFormation



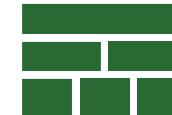
Framework

Stack creation

Stack updates

Error detection and rollback

Stack



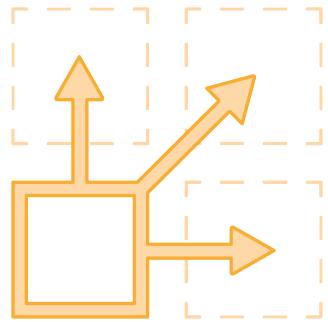
Configured AWS resources

Comprehensive service support

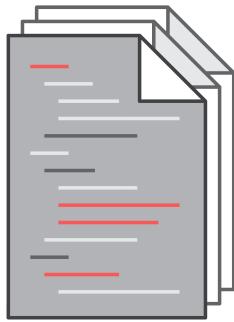
Service event aware

Customizable

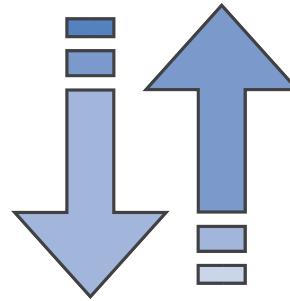
CloudFormation Benefits



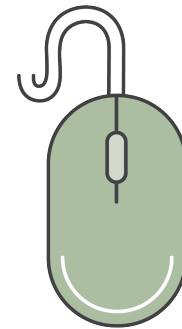
Templated resource provisioning



Infrastructure as code

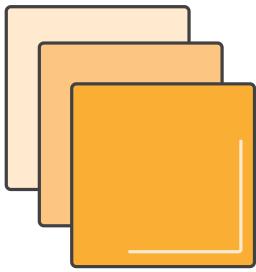


Declarative and flexible

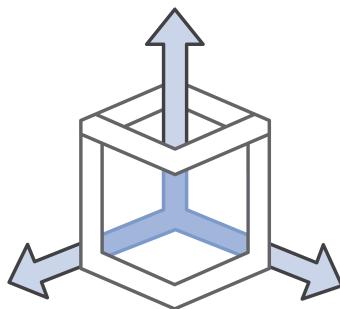


Easy to use

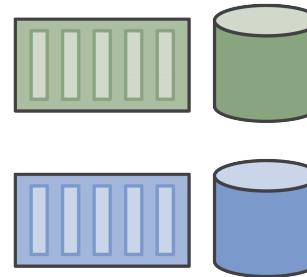
CloudFormation Use Cases



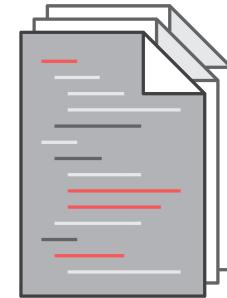
Stack replication



Infrastructure
scale out



Blue-green
deployments



Infrastructure
as code

Why do customers use CloudFormation?

Developers/DevOps teams value CloudFormation for its ability to treat infrastructure as code, allowing them to apply software engineering principles, such as SOA, revision control, code reviews, integration testing to infrastructure.

IT Admins and MSPs value CloudFormation as a platform to enable standardization, managed consumption, and role-specialization.

ISVs value CloudFormation for its ability to support scaling out of multi-tenant SaaS products by quickly replicating or updating stacks. ISVs also value CloudFormation as a way to package and deploy their software in their customer accounts on AWS.

EC2 Spot Instances

AWS EC2 Consumption Models

On-Demand

Pay for compute capacity by the hour with no long-term commitments

For spiky workloads, or to define needs



Reserved

Make a low, one-time payment and receive a significant discount on the hourly charge

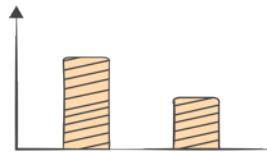
For committed utilization



Spot

Bid for unused capacity, charged at a Spot Price which fluctuates based on supply and demand

For time-insensitive or transient workloads



With Spot the rules are simple



Markets where the price of compute changes based on supply and demand



You'll never pay more than your bid. When the market exceeds your bid you get 2 minutes to wrap up your work

Show me the markets!

C3	1a	1b	1c	On Demand
8XL	\$0.50	\$0.27	\$0.29	\$1.76
4XL	\$0.21	\$0.30	\$0.16	\$0.88
2XL	\$0.08	\$0.07	\$0.08	\$0.44
XL	\$0.04	\$0.05	\$0.04	\$0.22
L	\$0.01	\$0.01	\$0.04	\$0.11

Each instance family

Each instance size

Each Availability Zone

In every region

Is a separate **Spot Market**

Spot Fleet



Spot fleet helps you



Launch Thousands of Spot Instances
with one `RequestSpotFleet` call.

Get Best Price

Find the lowest priced horsepower that works for you.
or

Get Diversified Resources

Diversify your fleet. Grow your availability.

And

Apply Custom Weighting

Create your own capacity unit based on your application needs

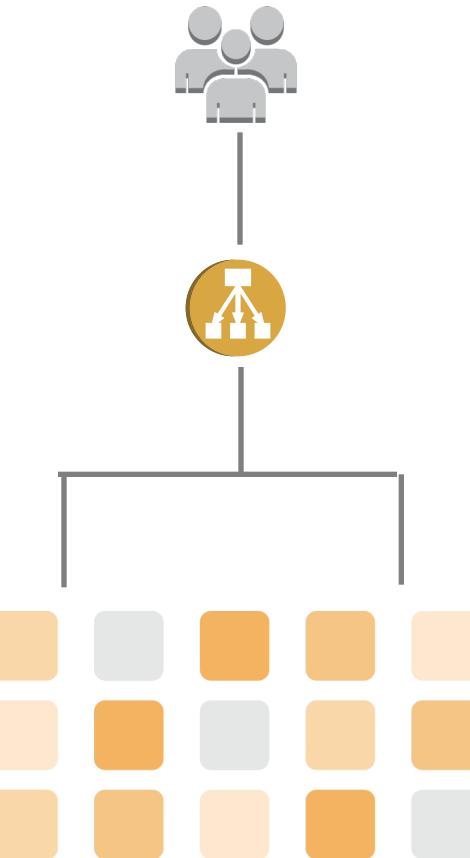
Diversification with EC2 Spot fleet



Multiple EC2 Spot instances selected

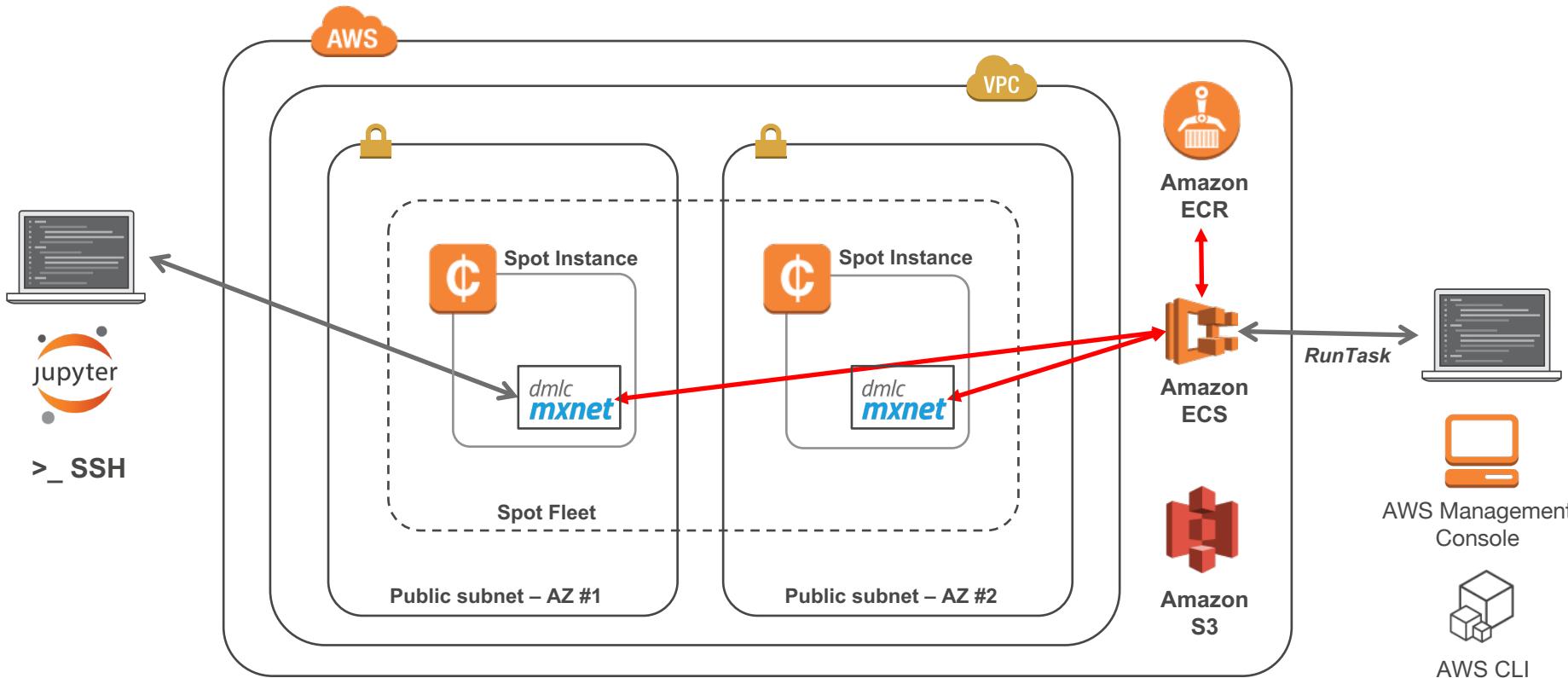
Multiple Availability Zones selected

Pick the instances with similar performance characteristics e.g. c3.large, m3.large, m4.large, r3.large, c4.large.

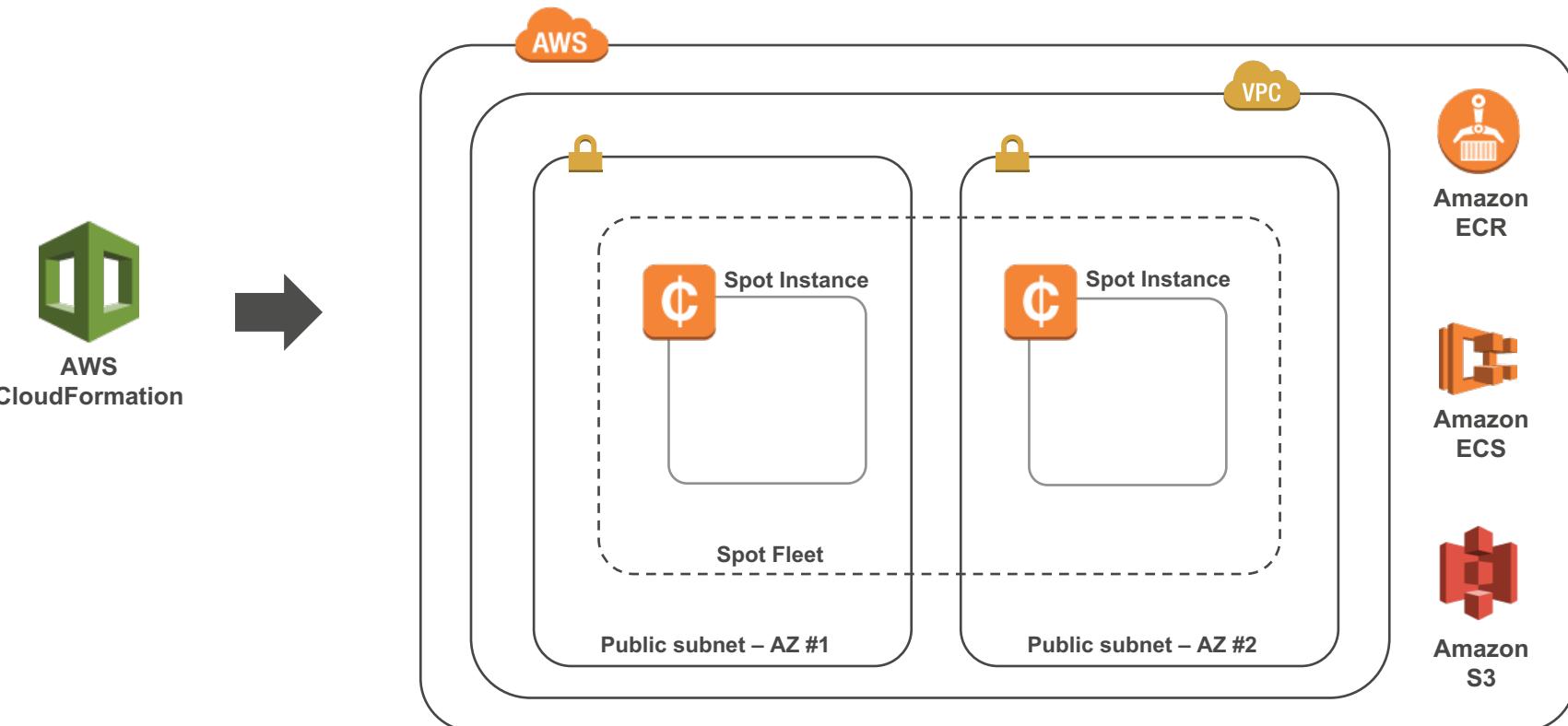


Workshop: Image Classification

Overall Architecture



Lab 1: Setup the Workshop Environment



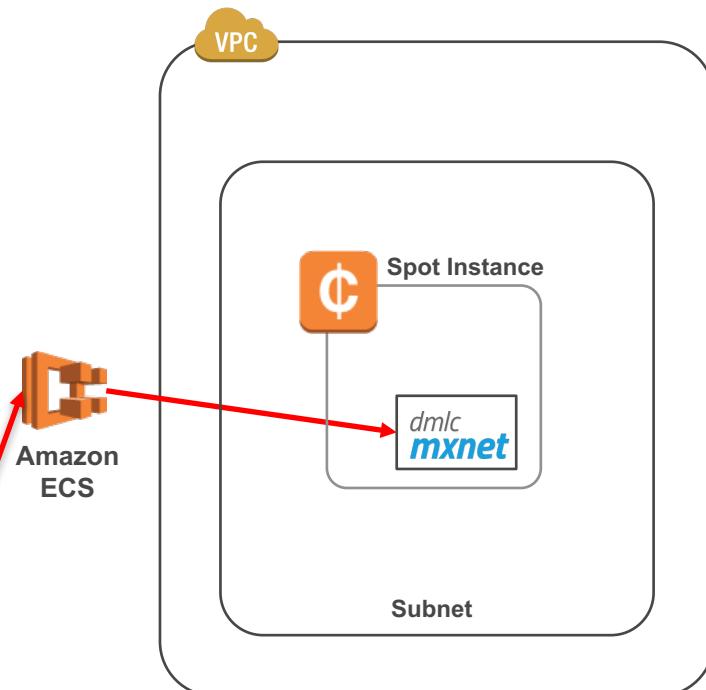
Lab 2: Build an MXNet Docker Image



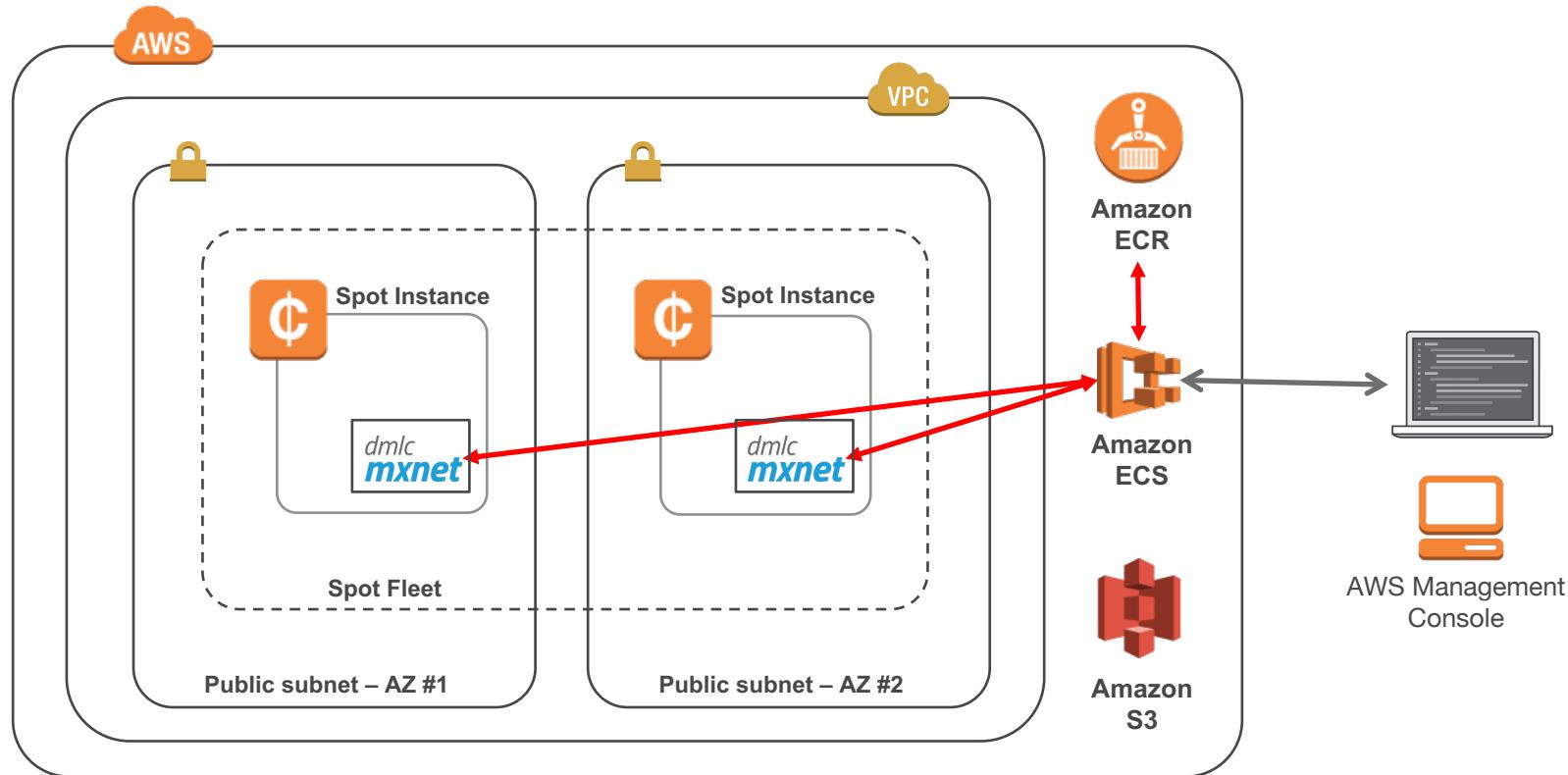
Amazon
ECR

```
"containerDefinitions": [
{
  "volumesFrom": [],
  "memory": 2048,
  "extraHosts": null,
  "dnsServers": null,
  "disableNetworking": null,
  "dnsSearchDomains": null,
  "portMappings": [
    {
      "hostPort": 80,
      "containerPort": 8888,
      "protocol": "tcp"
    }
  ],
  "hostname": null,
  "essential": true,
  "entryPoint": null,
  "mountPoints": [],
  "name": "mxnet",
  "ulimits": null,
  "dockerSecurityOptions": null,
  "environment": [],
  "links": null,
  "workingDirectory": null,
  "readonlyRootFilesystem": null,
  "image": "██████████ dkr.ecr.us-west-2.amazonaws.com/test:latest",
  "privileged": false
}]]
```

ECS Task Definition



Lab 3: Deploy MXNet Container with ECS



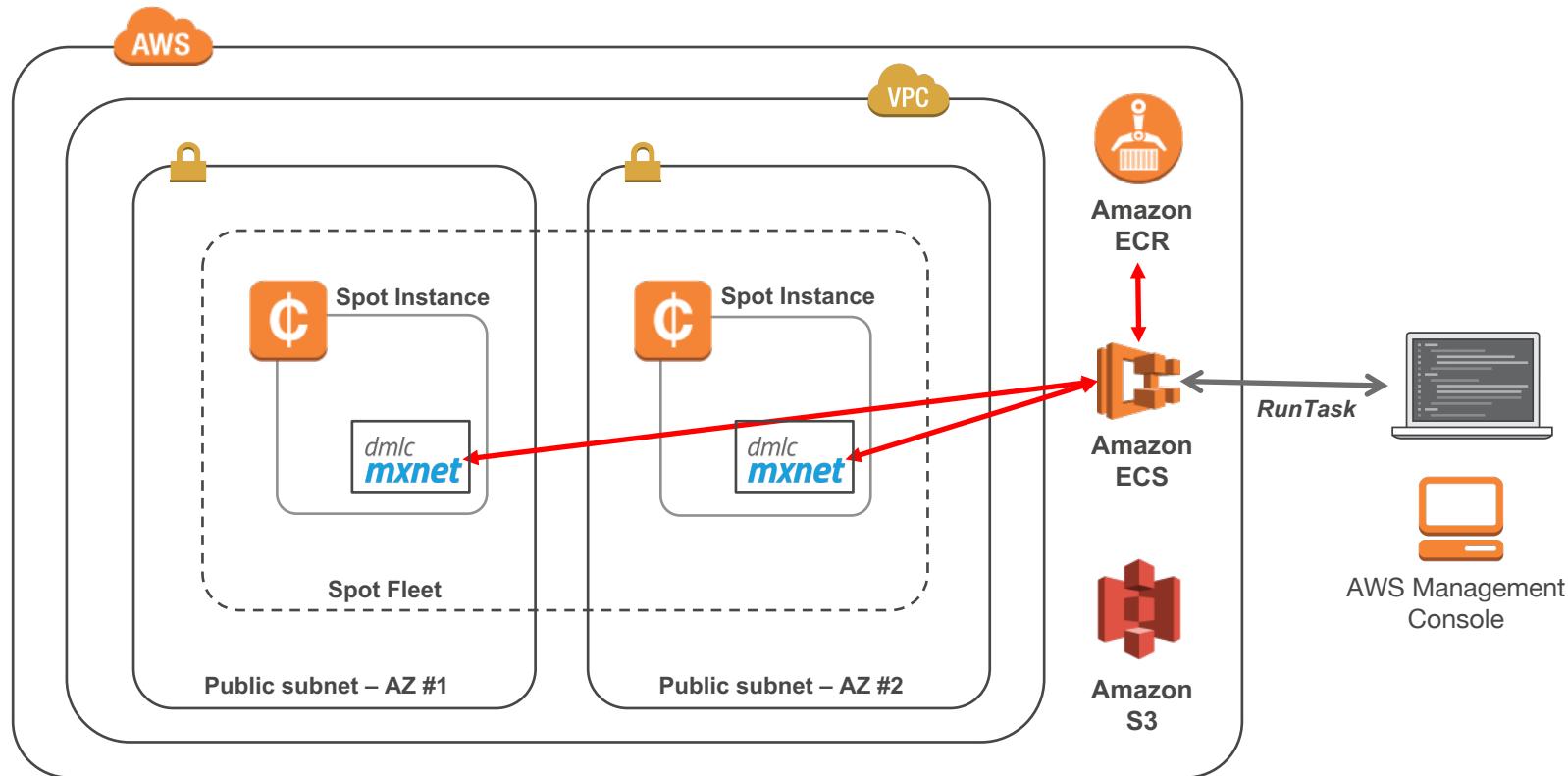
Lab 4: Image Classification Demo

```
In [4]: url = 'http://writm.com/wp-content/uploads/2016/08/Cat-hd-wallpapers.jpg'  
predict(get_image(url), mod, synsets)
```

```
probability=0.692329, class=n02122948 kitten, kitty  
probability=0.043847, class=n01323155 kit  
probability=0.030002, class=n01318894 pet  
probability=0.029693, class=n02122878 tabby, queen  
probability=0.026972, class=n01322221 baby
```



Lab 5: Wrap Image Classification in an ECS Task



Let's Get Started!

<https://github.com/awslabs/ecs-deep-learning-workshop>



Thank you!



**Remember to complete
your evaluations!**

Appendix

Related Sessions

- MAC306 - Using MXNet for Recommendation Modeling at Scale
- CON301 - Operations Management with Amazon ECS
- CON302 - Development Workflow with Docker and Amazon ECS
- CON401 - Amazon ECR Deep Dive on Image Optimization