Windows Containers on AWS
Ninja Bootcamp
Lab Guide (V1.0 R1.3)

Contents

Timing Tips

- **Plan your time carefully, you can easily run out of time**
- Skip optional labs (mandatory=No) and do them later if you have spare time.
- To save the time, try to do the labs while watching the videos. To do this, first scan through the question you need to answer and then start watching the video.
- You can watch the video at a higher speed (1.5x)/manually fast forward
- Execute labs starting with 1xx as quickly as possible. Do not spend too much time with docker basics.
- **Spend more time doing hands on activities**

Id	Title	mandatory	Instruction (min)	Estimated* time to complete(min)	
			Video time	Average user	Expert user
1	100_container_introduction	Yes	15	15	15
2	110_lab_setup	Yes	9	15	15
3	120_docker_basic_image_manipulation	Yes	13	30	20
4	130_docker_basic_run_containers	Yes	25	50	40
5	140_docker_basic_package_application	Yes	20	45	30
7	150_docker_basic_automate_image_build	No	11	30	20
6	160_docker_basic_resource_control	No	12	30	20
8	170_docker_basic_mount	No	12	30	20
9	200_ecr_ecs_introduction	Yes	10	10	10
10	210_ecr_publish_image	Yes	13	30	20
11	230_ecs_1_create_cluster	Yes	8	20	10
12	230_ecs_2_create_task	Yes	9	20	15
13	230_ecs_3_run_task	Yes	13	30	20
14	230_ecs_4_schedule_task	No	10	30	20
15	230_ecs_5_create_service	Yes	8	20	15
16	230_ecs_6_scaling_cluster	No	25	80	60
17	300_build_website_container	Yes	16	30	20
18	310_run_website_container	Yes	16	40	30
19	320_cicd_introduction	No	17	17	17
20	325_cicd_container_auto_deployment	No	30	60	60
Total – watch all videos and execute all labs (in hours)				**11**	**8**
Total – watch only mandatory videos and execute only mandatory labs (in hours)				**6**	**5**
Total – only watch mandatory videos and execute mandatory labs & only watch the videos related to optional labs (in hours)				**8**	**7**
Total – watch all videos and do no labs (in hours)				**5**	**5**
Total – watch only mandatory videos and do no labs (in hours)				**3**	**3**

Introduction

Welcome to Windows containers on AWS ninja bootcamp. In the next couple of hours, we are going to master Windows containers on AWS.

Lab documents and demos

Download the lab content (assets.zip) from following links and unzip the files to a folder <mylab> in your computer.

Description	Source	Destination
Scripts/sample code/programs	https://aws-ninja-bootcamp.s3-ap-southeast-2.amazonaws.com/ms-containers-on-aws/v-1-2/assets.zip	<mlab>\assets
Videos: if copied from *USB drive*	(only in special situations) - USB	<mlab>\videos

All videos & source files in this document are referred relative to <mylab> folder

If you do not have a tool to unzip, use 7zip https://www.7-zip.org/download.html

Now watch the video https://youtu.be/h1C2ihdYlz0

The video is also available at <mylap>\videos\100_introduction

If you already know container concepts, you can ignore this introduction

Important Changes: Mandatory Read

This learning series was first developed using Windows Server 2016. Microsoft has since released Windows Server 2019 with more bug fixes and feature upgrades to the docker. Although the videos were created using a Windows 2016 server, you will execute these labs on a Windows 2019 server and will use Windows 2019 base images.

Microsoft now uses its own container repository (mcr) instead of docker hub to distribute windows-based containers. Microsoft has also changed the naming convention of windows-based images. You can find more about these changes at https://techcommunity.microsoft.com/t5/Containers/Removing-the-latest-Tag-An-Update-on-MCR/ba-p/393045

Microsoft has updated the name of windows server core image from `microsoft/windowsservercore:latest` to **`mcr.microsoft.com/windows/servercore:ltsc2019`**. The videos still show the image name as `microsoft/windowsservercore:latest`. Please use the latest image name **`mcr.microsoft.com/windows/servercore:ltsc2019`** during this lab. For Windows containers, Amazon ECS by default creates Windows 2019 servers. Your lab environment AMIs are also upgraded into Windows 2019.

Lab Setup

In this session, we are going to create the lab environment. We will use this environment throughout the course.
Watch the video https://youtu.be/q_mwKn1YhyU
The video is also available at `<mylab>\videos\110_lab_setup`

If you are executing this lab environment in your own account or want to inspect the BasicLabSetup cloud formation template, you can find it at `<mylab>\assets\100_lab_setup\BasicLabSetup.template`

Execute following steps to login into your lab machine.

E 1. As given in the video, login into AWS console

Your instructor has given you the AWS account ID, username and password in an email sent to you or in a paper slip.

If you are executing the lab in an AWS provided accounts, your instructor has already provisioned a cloud formation template with basic network setup. You will have a VPC called contos-vpc-{some ids} and a few IAM roles and security groups. Since this is a container course, we will not spend our time learning thigs like VPCs & security groups.

E 2. As given in the video, create an EC2 instance (**`t2.large`**) in `AZ2 (10.0.4.0/24)` using the key pair have you created. AMI name(s) is given below.

- WindowsContainerWorkshop-V{x}

Please note that you need to click the check box "Shared with me" under "My AMIs". X will be changed time to time when we upgrade and patch the lab AMI.

If you are executing this lab on AWS provided accounts, you can only start t2 & t3 instances in small,medium,large sizes. We recommend t2.large or a t3.large instance. You have only limited permissions on this account. Retrieve the password using the keypair and RDP into the instance. Do the basic docker checks. If all works, you are ready to start the labs.

Docker Fundamentals

In this series of labs, you will learn basic Docker commands any container enthusiast must know. You need to know these commands by heart and more you use them, better you become at them. Do not expect a fancy UI to do these. The author used to be someone like you, who spent 7 years at Microsoft shops and expected fancy UIs to manage docker. He soon realized that world has changed and that Microsoft is moving towards scripting and PowerShell to manage servers. Once you learn the patterns, you will get hang of it.

1. Docker image manipulation commands

Watch the video: https://youtu.be/8Bw9ki6rEB0

The video is also available at `<mylab>\videos\ 120_docker_basic_image_manipulation`

You learned following commands

- `docker --help`
- `docker <sub command> --help`
- `docker info`
- `docker pull`

  `make sure you use the image` **`mcr.microsoft.com/windows/servercore:ltsc2019`** `instead of microsoft/windowsservercore:latest)`
- `docker images`
- `docker rmi`
- `docker image history`
- `docker image inspect`

Now answer following questions and do the exercises

E 1. Which public website you can find frequently used docker images?

E 2. Using command line, list all available images. Do the same, but with a different command.

E 3. Using command line, search for an image called `aspnet (docker search aspnet)`

E 4. Download an image called `mcr.microsoft.com/windows/nanoserver:1803` and tag the download image with a new name like `mycompany/mynanoserver:v1.0`.

E 5. Instead of `docker pull`, find another command to download an image

E 6. How can you remove all the content of the image you have download in Ex 4, so that the disk space used by that image is completely released?

E 7. In which folder docker images are kept in Amazon EC2 Windows instances? Browse to that folder and explore it

E 8. Imagine you downloaded an image. During docker run you don't specify a command to run. How do you know which program run by default when you start a docker container out of that image?

E 9. Your friend, Ishikawa, has built a docker image. You want to know which commands were executed to create that image. How do you find that? Find how **`mcr.microsoft.com/windows/servercore:ltsc2019`** `and` **`Microsoft/aspnet`** images were built

2. Docker run command

Watch the video https://youtu.be/Kui9k-8EBYs

The video is also available at `<mylab>\videos\130_docker_basic_run_containers`

You learned following commands

- `docker run -it <image name> <command to run>`
- `docker run -d <image name> <command to run>`
- `docker ps`
- `docker ps -a`
- `docker attach`
- `docker stop`
- `docker start`
- `docker rm`
- `docker exec`

Quickly scan through https://docs.docker.com/engine/reference/run/ (1 min max)

Now answer following questions and do the exercises to remember what you have just learned. Use the base image **`mcr.microsoft.com/windows/servercore:ltsc2019`** for all your activities.

E 1. Start PowerShell inside a container in interactive mode

E 2. Let's say you have executed the exercise E1. When you are inside the container (interactive mode) and want to know the container ID, how can you find it?

E 3. Can you terminate a process running inside the container from the container host? Start a continuous ping inside the container you have started in E1 by typing `ping 8.8.8.8 -t` and then stop it from container host.

E 4. When you are inside a container (let us say you start the container in interactive node with `docker run -it`) and want to exit from the container without stopping the container. Which key combination you should press? Try this with the container you have started in E1.

E 5. Let's say you have started a container in detached mode (`docker run -d`) or exit from the interactive mode as in E4. If you now want to get inside the container and go to interactive experience, how can you do that?

Go back to interactive experience, after you have executed exercise E4

E 6. Using command line, find which containers are still running and which containers have exited?

E 7. Your friend Ishikawa, started a few containers and went to have a good nap. You want to know which configurations were used when your friend started them (E.g. mount points, port mapping, and resource allocations, labels). How can you find the configurations for each running container?

E 8. Your friend Ishikawa started multiple containers and stopped them. He is on a vacation now. When you type `docker ps -a` you find a lot of stopped containers. How can you remove them one by one? Can you write a single PowerShell command line to clean all of them in one go (Tip: Google it)

E 9. Let's say you start a busy process in detached mode and the process is now running continuously (E.g. `docker run -t microsoft/windowsservercore ping 8.8.8.8 -t`). You now want to run another command on this container. How can you do that?

3. Package an exe applications

Watch the video https://youtu.be/ImWFA37QEt8

The video is also available at `<mylab>\videos\140_docker_basic_package_application`

You learned following commands

- `docker cp`
- `docker commit`
- `docker commit --change`

Now answer following questions and do the exercises to remember what you have just learned. Use the base image **`mcr.microsoft.com/windows/servercore:ltsc2019`** for all your activities.

E 1. As given in the video, create an image called `mybatchapp:latest`. The two sample exe files are available at

`<mylab>\assets\140_docker_basic_package_application\MyBatchApp.exe`

`<mylab>\assets\140_docker_basic_package_application\MySuperBusyApp.exe`

E 2. In this image, you need to have both `C:\myapp\MyBatchApp.exe` and `C:\myapp\MySuperBusyApp.exe`. We will use this image in subsequent labs. Therefore, do not remove the image after this exercise.

E 3. To verify that the image created in E 1 is working, start a container out of `mybatchapp:latest` and run `C:\myapp\MyBatchApp.exe` inside that

E 4. As given in the video, create an image called `abc` with `C:\myapp\MyBatchApp.exe` under `CMD attribute`

E 5. As given in the video create an image called `abc2` with `C:\myapp\MyBatchApp.exe` under `Entrypoint`

E 6. Clean the lab environment and remove the images `abc` and `abc2`. Keep the image `mybatchapp:latest`. we will use it in subsequent labs.

E 7. Can you save a running container as an image?

E 8. What is the difference between `Entrypoint` and `CMD` in an image?

4. Automating image build – Optional Lab

This is an optional lab. To save time, we strongly recommend you skip this to focus on more advanced topics. If you find enough time left, you can always do this later.

Image building can be complex. For example, you may need to execute complex commands & scripts to install a software and configure it. In this lab, you will learn how you can automate that process.

Watch the video https://youtu.be/Fn5ip4wTGek

The video is also available at `<mylab>\videos\150_docker_basic_automate_image_build`

You learned following commands

- `docker build`
- `docker build –tag <tag name>`
- `docker build --no-cache`

Now answer following questions and do the exercises to remember what you have just learned. Use the base image **`mcr.microsoft.com/windows/servercore:ltsc2019`** for all your activities.

E 1. Can you use a different file name for Dockerfile

E 2. Does the Dockerfile has an extension?

E 3. `<mylab>\assets\150_docker_basic_automate_image_build` has three Docker files.

Open them in notepad and explore the content. Observe how relative paths are defined. Copy them into your lab machine (one by one) renaming as Dockerfile. As given in the video, execute them with `docker build` to build different images. Remember to give a different tag for each image (E.g. `myweb:v1` , `myweb:v2`, `myweb:v3`, `myweb:v4`). The base images of these files are changed to **`mcr.microsoft.com/windows/servercore:ltsc2019`**

E 4. Using the new found knowledge, automate the creation of an image similar to `mybatchapp:latest` which we created in [3 Package an exe applications]-[E 1], however use a different name called `abc3:latest`.

E 5. We already learned about `CMD` and `Entrypoint` in previous lesson [3 Package an exe applications]. Can you think of a way to build `mybatchapp:latest` image using a docker file such that when the container runs, it runs `c:\myapp\MyBatchApp.exe` as the default command.

E 6. Using docker inspect, observe the differences between the images created with multiple commands vs images created with batching multiple commands into one.

E 7. What is the purpose of --no-cache option in docker build? Can you think of use case where caching is important?

5. Controlling resource allocation – Optional Lab

This is an optional lab. To save time, we strongly recommend you skip this to focus on more advanced topics. If you find enough time left, you can always do this later.

`MySuperBusyApp.exe` application uses too much CPU. Let's learn how you can allocate just enough resources to your containers.

Watch the video https://youtu.be/ihx_Bx-6sqk

The video is also available at `<mylab>\videos\160_docker_basic_resource_control`

You learned following commands

- `docker run --cpu-percent`
- `docker run --memory`

Now answer following questions and do the exercises to remember what you have just learned. Use the base image **`mcr.microsoft.com/windows/servercore:ltsc2019`** for all your activities.

E 1. Does Windows 2016 supports all the resource controls docker provides?

E 2. Quickly scan through the webpage

https://docs.microsoft.com/en-us/virtualization/windowscontainers/manage-containers/resource-controls

E 3. Using the image `mybatchapp:latest` you have built in [3 Package an exe applications]-[E 1], run `c:\myapp\MySuperBusyApp.exe` inside a container with 40% CPU and 200MB of memory allocation. Observe the actual resource utilization in the task manager. To prevent you waiting for a long time and to prevent your environment from crashing, allocate at least 200MB of memory.

6. Mounting folders & volumes – Optional Lab

This is an optional lab. To save time, we strongly recommend you skip this to focus on more advanced topics. If you find enough time left, you can always do this later.

In this lab, you will learn how you can share data and configuration across multiple containers by mounting folders and volumes to containers.

Watch the video https://youtu.be/ooauJ-a85Io

The video is also available at `<mylab>\videos\170_docker_basic_mount`

You learned following commands

- `docker run -v <source>:<destination mount location>`
- `docker volume`
- `docker volume ls`
- `docker volume rm`
- `docker volume create`

Now answer following questions and do the exercises to remember what you have just learned. Use the base image **`mcr.microsoft.com/windows/servercore:ltsc2019`** for all your activities.

E 1. As given in the video, create a folder in the container host and share it across two containers. Copy a file into that folder and see how it appears in the two containers. While inside the container, write a file into the mounted folder and observe how that file appears in the container host and in other containers.

E 2. Do you think it is a good practice to go to the folder where docker maintains volumes and delete the files manually? What is the proper way to delete a file from a docker volume?

E 3. Create a volume called `config` using docker commands. Mount that folder as `C:\app\myconfig` inside the container

E 4. Try to remove a volume, when a container (running or stopped) is using it. What sort of error you get? Remove all the containers that use the volume and then try to remove the volume. Do you get the same error?

E 5. What is the difference between bind mount and volume mount?

Amazon ECR & Amazon ECS

7. Amazon ECR & Amazon ECS introduction

Well done if you have completed all the previous labs. You now have a good knowledge about Windows containers.

AWS offers multiple container orchestration engines. These include…

1. Amazon Elastic Kubernetes Service
2. AWS Fargate
3. Amazon Elastic Container Service

Let us now focus on Amazon Elastic Container Registry (Amazon ECR) and Amazon Elastic Container Service (Amazon ECS).

Watch the video https://youtu.be/5bO4snSVsK0

The video is also available at `<mylab>\videos\200_ecr_ecs_introduction`

E 1. What is the advantage of using Amazon Elastic Container Registry over a repository like docker hub?

E 2. Why do you need a service like Amazon ECS?

8. Publish a docker image to Amazon Elastic Container Registry (Amazon ECR)

In this lab, we will learn how to publish our image (`mybatchapp:latest`) to Amazon ECR. We created this image in a previous lab [3 Package an exe applications]. If you haven't done it, please execute that lab and get your `mybatchapp:latest` image created in your lab machine.

Watch the video https://www.youtube.com/watch?v=89gy31sWdOs https://youtu.be/IZNcim4M4Rg

The video is also available at `<mylab>\videos\210_ecr_publish_image`

You learned following commands. As given in the video, you can find the appropriate commands in Amazon ECR console to copy & paste.

- `aws ecr get-login`
- `Invoke-Expression -Command (Get-ECRLoginCommand … ).Command`
- `docker push`

Now answer following questions and do the exercises to remember what you have just learned

E 1. As given in the video, create a repository called `mybatchapp`

E 2. As given in the video, explore the IAM role (`My-Container-Host-Role`) attached to your lab machine and check whether it has the managed permission `AmazonEC2ContainerRegistryFullAccess` attached to it.

E 3. Can you give more fine grain access to your Amazon ECR repository with custom permission? (Yes|No)

E 4. Think about use cases where you want to grant production servers read access to the repository and staging/ build servers write access to the repository.

E 5. Quickly scan through (30 second read)

https://docs.aws.amazon.com/AmazonECR/latest/userguide/logging-using-cloudtrail.html

E 6. As given in the video, using PowerShell, login into `mybatchapp` repository you created in E 1. You should get Login Succeeded Message.

E 7. As given in the video, tag `mybatchapp:latest` image to point to the full URL of the Amazon ECR repository you have created in E1

E 8. As given in the video, publish the `mybatchapp:latest` image to the repository you have created in E1

E 9. Can you maintain multiple versions/tags images in a single Amazon ECR repository?

E 10. In Amazon ECR, can you have multiple tags that point to the same image content? What's the use case for this?

9. Create an Amazon ECS cluster

Watch the video https://www.youtube.com/watch?v=ODF0-HCzIKk https://youtu.be/9l17yY09VTU

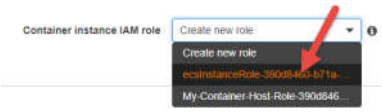
The video is also available at `<mylab>\videos\230_ecs_1_create_cluster`

Now answer following questions and do the exercises to remember what you have just learned

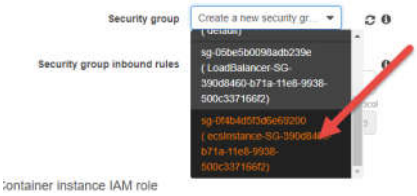
E 1. As given in the video, create "EC2 Windows +Networking Amazon ECS cluster" named MyContosoCluster. **Important readings before you create a cluster:**

Select an instance type of t2.large and create a cluster of only one instance. {*Note: Do not select big instances, If you are executing this lab on AWS provided accounts, you can only start t2 & t3 instances in medium,large,xlarge sizes}*. Since this is a lab environment, there is limited AWS credits available. If you are running this lab in AWS provided accounts, you can only select on demand instances. You do not have permission to run AWS spot instances.

Make sure you select the correct IAM role. Cluster instances have to have correct permission so that they can communicate with Amazon ECS and Amazon ECR. Initial lab setup has provisioned an IAM role named `ecsInstanceRole-{id}` (E.g. ecsInstanceRole-390d8460-b71a-11e8-9938-500c337166f2). As shown in the video, use this role as the IAM role for Amazon ECS instances instead of creating a new one.



Initial lab setup (CloudFormation: BasicLabSetup) has provisioned a security group named `ecsInstance-SG-{id}` (E.g. ecsInstance-SG-390d8460-b71a-11e8-9938-500c337166f2}. Use this security group instead of creating a new one. This security group allows traffic from the load balancer which we will create in later labs.



Since we are going to remote login into the "Amazon ECS cluster instance" to explore what is happening behind the scene, remember to select a key pair. In production environments, you do not define a key pair. We run this cluster in public subnets so that we can easily execute the labs without complex setup. However, in production environments you may run this cluster in private subnets.

In this lab series, we are exploring only unmanaged EC2 based clusters so that you learn all the internals and how Amazon ECS works. AWS also has a managed cluster service called AWS Fargate. You may find UI elements related to this. Please ignore AWS Fargate related UIs and select only EC2 based cluster.

E 2. As given in the video, observe the CloudFormation execution and check which AWS resources got provisioned as part of the cluster creation. Navigate around EC2 section of the AWS console & observe following resources that got created

- EC2 launch configuration
- Auto scaling group
- EC2 instance started by the auto scaling group

E 3. How much CPU units and memory (MB) the newly added EC2 instance provides to the cluster?

10. Create an Amazon ECS task definition

We will now create an Amazon ECS task definition to point into Amazon ECR repository.

Watch the video https://youtu.be/MEKtOWTYMp8

The video is also available at `<mylab>\videos\230_ecs_2_create_task`

Now answer following questions and do the exercises to remember what you have just learned

E 1. As given in the video, create a task definition called `MyBatchAppTask`.

Notes:

The version you get for your task may be different from what you see in the video.

Although you can give your own task name, we recommend you keep the same name `MyBatchAppTask`, because subsequent videos will then become easy to follow.

In the command section, under container definition, do not use `c:\myapp\MySuperBusyApp.exe`. It will quickly drain your t2 instance credit balance and you will experience a slow cluster. As given in the video, use `c:\myapp\MyBatchApp.exe` in the task definition command.

Do not select **Enable App Mesh integration or Enable proxy configuration**

E 2. If you have time, quickly scan through following links (max 1 min each)

https://docs.aws.amazon.com/AmazonECS/latest/developerguide/application_architecture.html

https://kubernetes.io/docs/concepts/workloads/pods/pod/

11. Run an Amazon ECS task on the cluster

Watch the video https://youtu.be/fGaFaWvSN6c

The video is also available at `<mylab>\videos\230_ecs_3_run_task`

Now answer following questions and do the exercises to remember what you have just learned. If you are coming from Kubernetes background a task is similar to a Kubernetes pod.

E 1. As given in the video, run **one** instance of the task you have created in [10 Create an Amazon ECS task definition]-[E 1]

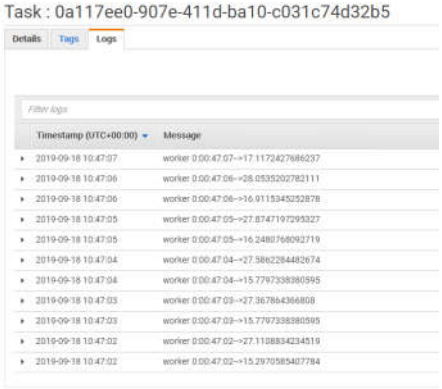
When you run a task for the first time, the container host may not have the images for that task. You may need to wait a bit for the task to start (~5-10min). This delay compromise of

- The time lag between you scheduling the task and Amazon ECS agent picking that task
- The time it takes to download the initial image and extracting it on the container host

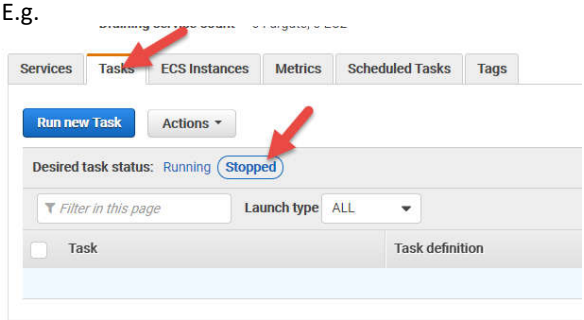
However, subsequent starts of the task should be fast. In production environments, the container host is configured to have the main layers, so the download and extract phases complete quickly. If you feel like your task never runs, you may have configured the task or container host incorrectly. Common errors are…

- Wrong task definition (E.g. pointing to a CloudWatch log group that does not exists, wrong image name)
- Container host have no permission (E.g. container host does not have permission to read from Amazon ECR or to communicate with Amazon ECS APIs)
- Security groups or network ACL blocking the traffic

If you task unusually starts and stops, very likely you have configured it wrongly. Explore task log to isolate any errors. You can find console output of the batch file we are running from the logs. E.g.



If you task fails prior to starting the container, you can look at stopped task to identify possible errors like wrong image name. E.g.



E 2. As given in the video, to understand what happens behind the scene when a task is running, remote login into the EC2 instance on which the task is scheduled and execute a few docker commands to inspect the docker images and running containers. Note that in production environments we do not do this.

ECS agents keeps its logs at C:\ProgramData\Amazon\ECS\log

This is a great place to find common errors. For example, if your container host does not have permission to read from Amazon ECR you will find those errors in these logs.

E 3. As given in the video, stop the container from PowerShell and observe what happens to the task running in Amazon ECS. Note that in production environments we do not do this.

E 4. As given in the video, start another execution of the task and stop it from the Amazon ECS console. Using PowerShell observe what happens behind the scene.

E 5. As given in the video, observe how CloudWatch logs captures the console output of the batch application you are running.

Why do you think it is crucial to capture logs into a central location like Amazon CloudWatch?

E 6. As given in the video, try to run more tasks than what the cluster can support. What will be the outcome? If the cluster can support resource demands for two `MyBatchAppTasks` and currently running only one such task, what will happen if you start 3 additional `MyBatchAppTasks`?

12. Schedule Amazon ECS tasks – Optional Lab

This is an optional lab. To save time, we strongly recommend you skip this to focus on more advanced topics. If you find enough time left, you can always do this later.

In this lab, we will setup tasks to automatically start on the cluster.

Watch the video https://youtu.be/tF6ZiSpJuKw

The video is also available at `<mylab>\videos\230_ecs_4_schedule_task`

Now answer following questions and do the exercises to remember what you have just learned

E 1. As given in the video, create a schedule to start `MyBatchAppTask` every one minute.

E 2. As given in the video, observe how a rule get created in CloudWatch to trigger a task execution

E 3. Do you think it is normal for a task to run without terminating? (Yes| No| It depends)

E 4. As given in the video, reduce the CPU demand of the task definition to 128 CPU units and update the task. Also, update the schedule job to point to the new version of the task. Stop the running old version of the tasks and observe how many tasks will be scheduled on the cluster.

What is the value of having versions in the task definition?

E 5. As given in the video, clean the lab environment by deleting any schedules you have setup and stopping all the running tasks. This will prevent you running out of T2 instances credit balances.

13. Create an Amazon ECS service

In this lab, we will create a service which can maintain a desired number of tasks on the cluster.

Watch the video https://youtu.be/hDFegHtENu4

The video is also available at `<mylab>\videos\230_ecs_5_create_service`

As given in the video, remember to modify `MyBatchAppTask` definition to use 256 of CPU units and 256 MB of memory before executing this lab.

Now answer following questions and do the exercises to remember what you have just learned

E 1. Why you need a concept call service in the first place?

E 2. As given in the video, create a service called `MyBatchService` with desired count of four, and observe how it automatically starts four tasks.

E 3. As given in the video, to simulate a failure or a disaster, stop two of the running tasks and observe how the service automatically starts additional tasks to bring the running count equals to desired count.

E 4. As given in the video, observe the `Event` section of the service and check what events are logged when E 3 is executed.

E 5. As given in the video, to clean up the lab environment, stop the running tasks started by the service.

Note that just forcibly stopping the tasks will not work, because the service is going to start additional tasks to bring running count equals to desired count. You can either set the desired count of the service to zero or completely delete the service. The proper method is to set the desired count to zero. If you keep your tasks running, they can easily drain your T instance credit balances.

14. Horizontally scaling the cluster – Optional Lab

This is an optional lab & takes a disproportionate amount of time to execute. To save time, we strongly recommend you skip this to focus on more advanced topics. If you find enough time left, you can always do this later.

So far, we have used only a single Amazon EC2 instance in our cluster. In this lab, we will explore how we can add additional capacity to our cluster.

Watch the video https://youtu.be/ps8rP2tq3SA

The video is also available at `<mylab>\videos\230_ecs_6_scaling_cluster`

As given in the video, remember to modify `MyBatchAppTask` definition to use 256 of CPU units and 256 MB of memory before executing this lab.

To be frugal on our lab environment accounts, do not add more than 2 Amazon EC2 instance to the cluster and select only small instances as given in the video.

Now answer following questions and do the exercises to remember what you have just learned

E 1. Do the following calculation: If you are using four c5.2xlarge instances in the cluster, and you are going to start a task definition that demands 512 CPU units and 256 MB of memory, how many of those tasks can be started on the cluster.

Refer to the number of vCPUs and amount of memory in different instance families/types at https://aws.amazon.com/ec2/instance-types/

E 2. As given in the video, first manually add an additional instance to the cluster so that the total instances in the cluster is 2.

E 3. As given in the video, manually remove the instance added at E2 by setting the auto scaling parameters either through Amazon ECS UI or through EC2 auto scaling UI. At the end of this, you should have only one EC2 instance registered against the cluster.

E 4. Keep only two running `MyBatchAppTasks` tasks.

As given in the video, create a scaling policy, define a cloud watch alarm and set the cluster to automatically scale out when the cluster CPU reservation percentage reaches above 50%.

Make sure the auto scaling is configure with max instances equals to 2, so that when the scaling out even triggers, auto scaling group can spin up a new instance.

Once configured, start additional tasks and bring the cluster CPU reservation above 50%. Wait for a few minutes to see how the cluster automatically adds additional capacity.

E 5. Can you use custom CloudWatch matrices (E.g. total number of items in shopping basket or Madagascar GDP value) to scale the cluster in and out? (Yes| No)

E 6. As given in the video, remember to clean the lab environment by setting the max & desired count of the auto scaling group to one so that the cluster will have only one Amazon EC2 instance. This will prevent you from running out of AWS credits allocated to your lab environment.

Containerizing Web Applications

15. Containerizing an ASP.NET web application

Important Changes (Mandatory Read):

- Since the videos have been released, we have upgraded lab environment to have **Visual Studio Community Edition 2019** (Instead of 2017).
- Sample MyWebApp's base docker image is changed to **`mcr.microsoft.com/dotnet/framework/aspnet:4.8`** (Instead of **microsoft/aspnet:4.7.2**). Use the base image **`mcr.microsoft.com/dotnet/framework/aspnet:4.8`** `for this lab.`

Up until now, we have containerized and used only exe/batch applications. You can containerize other types of applications such as IIS web applications and windows services. In this lesson, we are going to containerize an ASP.NET web application. You do not need to be a developer to do this lab.

Watch the video https://youtu.be/M7Hoeb4DI-E

The video is also available at `<mylab>\videos\300_build_website_container`

Your lab machine has free Visual Studio Community Edition 2019 installed. You can open the ASP.Net website from it. If it prompts for license key, you can login with a Microsoft credentials (Hotmail, outlook email address) and the free edition will be unlocked.

Now answer following questions and do the exercises to remember what you have just learned

E 1. As shown in the video, pull the image **`mcr.microsoft.com/dotnet/framework/aspnet:4.8`** to your lab machine

E 2. A sample website is given at `<mylab>\assets\300_build_website_container`

As given in the video, copy the folder `MyWebApp` to your lab machine, open Visual Studio as an administrator and open the Visual Studio solution file `MyWebApp\MyWebApp.sln`.

Build & run the website in debug mode and view the web application appearing on your lab machine's browser.

Can you access the website with `localhost` instead through the special NAT IP the container got?

E 3. Check the new image that got created as part of E 2 execution. What is the name and tag of that image?

E 4. Check the new container Visual Studio has started as part E 2 execution. Can you find the special NAT IP address of the running website by inspecting the running container using PowerShell? What is the IP address your container got?

E 5. Quickly scan through following URL (1 min max)

https://docs.microsoft.com/en-us/virtualization/windowscontainers/container-networking/network-drivers-topologies

E 6. What's the program/(s) running in the container you started in E2

E 7. As given in the video, click the debug stop button in Visual studio to stop debugging the application. Does this action stop the container Visual Studio has started?

E 8. As given in the video, select **release build** and build the project to create a release build of the image. What is the name and tag of the image that got created?

E 9. Inspect the two images `mywebapp:dev` (which got created as part of E 2) and `mywebapp:latest` (which got created as part of E 8). What is the difference between Entrypoint for those two images? What's the purpose of the extra program, other than IIS, that is configured in `mywebapp:dev` Entrypoint

E 10. As given in the video, run the image `mywebapp:latest` using PowerShell. Find out which special NAT IP address your container received. Since you now know the IP address, using your browser, browse to that website.

E 11. In our previous labs, we learned how to create Amazon ECR repositories and publish images to it. As given in the video, create an Amazon ECR repository called `mywebapp` and publish the image `mywebapp:latest` to it. We will use this repository and the published image in subsequent labs.

E 12. To do a little bit of clean up, close Visual Studio, and stop all the running containers in your lab machine.

16. Run an ASP.NET web application on Amazon ECS

In the lab [0 Containerizing Web Applications

Containerizing an ASP.NET web application] we created a docker image that has an ASP.NET web application. We published this container to Amazon ECR repository called `mywebapp`. In this lab, we are going to learn how to run this web application on Amazon ECS.

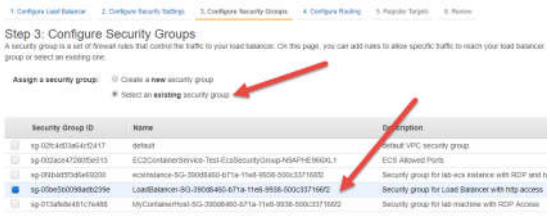
Watch the video https://youtu.be/R72a2hVBel0

The video is also available at `<mylab>\videos\310_run_website_container`

Now answer following questions and do the exercises to remember what you have just learned

E 1. As given in the video, create a task called `MyWebAppTask`. Make sure you allocated at least 1024 units of CPU and 1024 MB of memory. Without those resources, you will not be able to run your containerized web application in IIS.

As given in the video, create an application load balancer. However, instead of creating a new security group, select an existing security group called LoadBalancer-SG-{id} (E.g. LoadBalancer-SG-390d8460-b71a-11e8-9938-500c337166f2) that got created as part of initial lab setup (E.g. CloudFormation:BasicLabSetup). Note that Amazon ECS instances' security groups ecsInstance-SG-{id} allows traffic from LoadBalancer-SG-{id}. *If you don't have such traffic flow, your containers will start but the load balancer will never be able to connect to them. Load balancer will think containers are unhealthy and will stop them. The Amazon ECS service will then start the containers to bring the service to the desired state. You will find containers starting and stopping.*



E 2. Create an Amazon ECS service called `MyWebAppService` and run two tasks under it.
Note that service discovery is not supported for <default> network mode. So do not enable service discovery for this web application.

Configure `MyWebAppService` to use the load balancer. Once the service starts, using the DNS name of the application load balancer, access the website. Refresh the webpage to see how the load balancer does round robin load balancing between the two containers. Every time when you refresh the website, you will get a new container ID (you may need to use a browser like FireFox).

E 3. As given in the video, can you find which IP addresses and ports the containers are running?

E 4. If you are planning to execute the next labs (CI/CD), you don't need to clean up the lab environment. We need the service `MyWebAppService` in the next lab. Otherwise, you can delete the service `MyWebAppService`.

Continuous Integration and Continuous Delivery (CI/CD)

17. Continuous Integration and Continuous Delivery (CI/CD) introduction

Watch the video https://youtu.be/uJps5d5EJvU

The video is also available at `<mylab>\videos\320_cicd_introduction`

You now have some understanding of container deployment using a CI/CD pipeline.

Now answer following questions to remember what you have just learned

E 1. Are AWS service APIs exposed as PowerShell (Yes | No)

E 2. Are AWS service APIs exposed as command line (Yes | No)

E 3. Are AWS service APIs exposed in .NET (Yes | No)

E 4. Can you integrate your on-premises build system like Jenkins/Microsoft Team Foundation Server (TFS)/Auzre DevOps with AWS?

E 5. Can a build agent run multiple build jobs? (Yes | No)

18. Automating container deployment

Watch the video https://youtu.be/PR9Qo6sIP7g

The video is also available at `<mylab>\videos\325_cicd_container_auto_deployment`

Now answer following questions and do the exercises to remember what you have just learned

E 1. The automation PowerShell script `CICDAutomation` is given at
`<mylab>\assets\325_cicd_container_auto_deployment`

As given in the video, copy the file `CICDAutomation` to your lab machine, open PowerShell editor as an administrator, and open the script you have just copied. Read the script and try to understand what each step is doing.

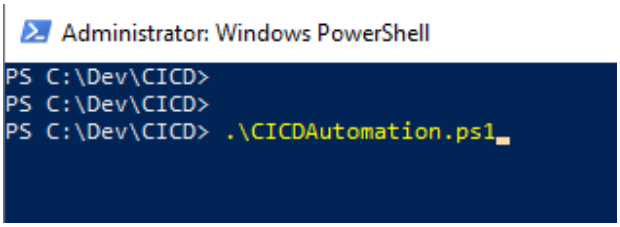
As given in the video, make modification to the script to make it suitable to your environment.

E.g. You may need to change the Visual Studio `MyWebApp` solution location, Amazon ECS service name and Amazon ECR repository name to suit the names you have given. Make sure you point the MSBuild.exe to a valid MSBuild file.

When you try to execute the script using the editor's play button you may get an error similar to following.

```
Trying to login to ECR with EC2 instance credentail
docker : WARNING! Using --password via the CLI is insecure. Use --password-stdin.
At line:1 char:1
+ docker login --username AWS --password eyJwYXlsb2FkIjoiUHk4SyszRWo5QT ...
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : NotSpecified: (WARNING! Using ...password-stdin.:String) [], RemoteException
    + FullyQualifiedErrorId : NativeCommandError
```

To avoid this error, you can directly execute the modified script with PowerShell. Open PowerShell as an administrator and go the folder where you copy and modify the CICDAutomation.ps1 file. Then execute the script by typing .\CICDAutomation.ps1. E.g.



E 2. As given in the video, make modification to your service desired count to one and stop any other running tasks, so that multiple deployments can work and will not fail/stuck due to lack of cluster resources.

Run the script and observe how the web application got built, containerized, published and deployed to Amazon ECS in one shot. As given in the video, observe how changes are propagating across different components.

E 3. As given in the video, make a modification to the `index.cshtml` file to simulate a developer making a change to a feature. Run the script again to automatically build, containerize, publish and deploy the image to Amazon ECS. Observer how you get the new website with minimal downtime. You may need to wait a bit until the load balancer passes the heath checks and starts redirecting the traffic to new containers. As given in the video, observe how changes are propagating across different components.

Clean up

19. Clean up

To be nice to AWS instructors, clean up your temporary AWS accounts following these instructions

-Delete any running Amazon ECS services

-Stop all running Amazon ECS tasks

-Delete Amazon ECS cluster MyContosoCluster (or any other clusters you have created)

-Delete all the Load Balancer

-Delete all auto-scaling groups

-Terminate all the Amazon EC2 instances

-Delete any left over EBS volumes (if any)

-Delete any left over Network interfaces (if any)

-Delete any public IP addresses (if any)

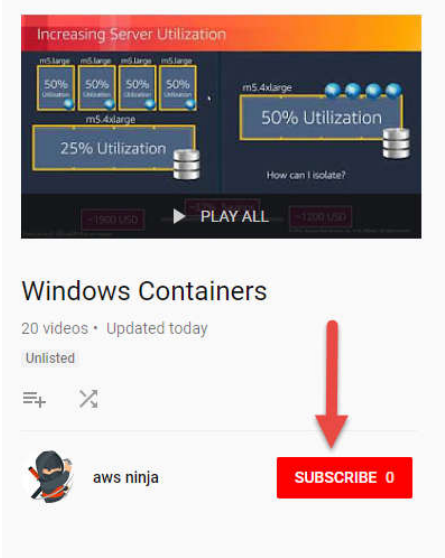
-Delete Amazon ECR repositories you have created

Once done, make sure you do not have any running/stopped Amazon EC2 instances.

List of videos

All videos are available as a playlist https://www.youtube.com/playlist?list=PLDvuRFHx48vAUKebX5sQa2nYSTeUS2Mh5

Feel free to subscribe to the YouTube channel for deep dive hands-on technical content



You now have tools and knowledge to containerize your Microsoft workloads and run them on AWS

Well done warrior!

Version 1.0