□□□□□□AWS□□

□□ □□ (Tomoyuki Mano) <tomoyukimano@gmail.com>

Version 1.0, 2021-06-14

Table of Contents

1. □□□□ . . . 2
  1.1. □□□□□□□□ . . . 2
  1.2. □□□□□□□□□□ . . . 2
  1.3. AWS□□□□□ . . . 3
  1.4. □□□□ . . . 3
  1.5. □□□□ . . . 4
  1.6. □□□□□□□□□ . . . 4
  1.7. □□□□□□□□□□□□□□□□ . . . 4
2. □□□□□□ . . . 6
  2.1. □□□□□□□ . . . 6
  2.2. □□□□□□□□□□□□ . . . 8
3. AWS□□ . . . 11
  3.1. AWS□□□ . . . 11
  3.2. AWS□□□□□□□□ . . . 11
  3.3. Region □ Availability Zone . . . 13
  3.4. AWS□□□□□□□□ . . . 14
  3.5. CloudFormation □ AWS CDK . . . 18
4. Hands-on #1: □□□□EC2□□□□□□□□□□□ . . . 22
  4.1. □□ . . . 22
  4.2. SSH . . . 22
  4.3. □□□□□□□□□□□ . . . 23
  4.4. □□□□□□□□□□□ . . . 27
  4.5. □□ . . . 33
5. □□□□□□□□□□□□□□□□ . . . 34
  5.1. □□□□□□□□□□□□□□□□□ . . . 34
  5.2. GPU □□□□□□□□□□□ . . . 34
6. Hands-on #2: AWS □□□□□□□□□□□□□ . . . 38
  6.1. □□ . . . 38
  6.2. □□□□□□□□□□□ . . . 38
  6.3. □□□□□□□□□□ . . . 41
  6.4. □□□□ . . . 42
  6.5. Jupyter Notebook □□□ . . . 44
  6.6. PyTorch□□□□□ . . . 46
  6.7. □□□□□□□□□□□! MNIST□□□□□□□□□□ . . . 49
  6.8. □□□□□□□ . . . 54
7. Docker □□ . . . 56
  7.1. □□□□□□□□□ . . . 56
  7.2. Docker □□ . . . 57

7.3. Docker □□□□□□□ . . . 59
7.4. Elastic Container Service (ECS) . . . 62
8. Hands-on #3: AWS □□□□□□□□□□□□□□□□ . . . 64
8.1. Fargate . . . 64
8.2. □□ . . . 65
8.3. Transformer □□□□ question-answering □□□□□ . . . 65
8.4. □□□□□□□□□□□□ . . . 67
8.5. □□□□□□□□□□ . . . 71
8.6. □□□□□□□ . . . 73
8.7. □□□□□□□□□ . . . 74
8.8. □□□□□□□□ . . . 76
9. Hands-on #4: AWS Batch □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ . . . 77
9.1. Auto scaling groups (ASG) . . . 77
9.2. AWS Batch . . . 77
9.3. □□ . . . 79
9.4. MNIST □□□□□□□□ (□□) . . . 79
9.5. □□□□□□□□□□□□ . . . 82
9.6. □□□□□□□□□□ . . . 86
9.7. Docker image □ ECR □□□□□ . . . 88
9.8. □□□□□□□□□□□□ . . . 91
9.9. □□□□□□ Job □□□□□ . . . 95
9.10. □□□□□□□□ . . . 99
9.11. □□ . . . 100
10. Web □□□□□□□□□ . . . 102
10.1. □□□□□□□□□□□ — Twitter □□□ . . . 102
10.2. REST API . . . 103
10.3. Twitter API . . . 104
11. Serverless architecture . . . 106
11.1. Serverful □□□□ (□□□) . . . 106
11.2. Serverless □□□□□ . . . 107
11.3. □□□□□□□□□□□□□□□□□□□□□□□□□□ . . . 108
12. Hands-on #5: □□□□□□□□□ . . . 113
12.1. Lambda □□□□□ . . . 113
12.2. DynamoDB □□□□□ . . . 118
12.3. S3 □□□□□ . . . 122
13. Hands-on #6: Bashoutter . . . 126
13.1. □□ . . . 126
13.2. □□□□□□□□□□□□ . . . 127
13.3. □□□□□□□□□□□□□□ . . . 134
13.4. API □□□□□□□□□□□ . . . 136
13.5. □□□ API □□□□□□□□□□□□□□□ . . . 138

13.6. Bashoutter GUI □□□□□□□ 139
13.7. □□□□□□□□□□□ 140
13.8. □□ 141
14. □□□ 142
15. Appendix: □□□□ 143
15.1. AWS □□□□□□□□ 143
15.2. AWS □□□□□□□□□□□□ 147
15.3. AWS CLI □□□□□□□ 149
15.4. AWS CDK □□□□□□□ 150
15.5. WSL □□□□□□□ 151
15.6. Docker □□□□□□□ 154
15.7. Python venv □□□□□□□ 155
15.8. □□□□□□□□□ Docker image □□□□ 156
16. □□ 158
17. □□□□ 159
18. □□□□□ 160

□□□

Chapter 1. □□□□

1.1. □□□□□□□□□

□□□□□□□□□□□□□□□2021□□S1/S2□□□□□□□□□□□"□□□□□□□□□□□"□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□

□□□

- **UNIX □□□□□□**: □□□□□□□□□□□□□□□□□□□□□□□□ SSH □□□□□□□□□□□□□□□□ UNIX □□□□□□□□□□□□□□□□□□ Mac □□□ Linux □□□□□□□□ OS □□□□□□□□□□□□(□□□□□□□□□□□□)□□□□□□□□□□ Windows □□□□□□□□ Windows Subsystem for Linux (WSL) □□□□ Linux □□□□□□□□□□□□□□□□□□□ (Section 15.5 □□)□
- **Docker**: □□□□ Docker □□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□ Section 15.6 □□□□□□□□
- **Python**: Version 3.6 □□□□□□□□□□□□□ □□□□□□□□□□□□□ `venv` □□□□□□□□□□□□□ `venv` □□□□□□ Section 15.7 □□□□□□□
- **Node.js**: version 12.0 □□ □□□□□□□□□□□
- **AWS CLI**: Version 2 □□□□□□□□□□□ □□□□□□□□□□□□□□□ Section 15.3 □□□□□□□
- **AWS CDK**: Version 1.100 □□□□□□□□□□□□□ Version 2 □□□□□□□□□□□□ □□□□□□□□□□□□□□□ Section 15.4 □□□□□□□
- **AWS □□□□□□**: AWS API □□□□□□□□□□□□□□□□□□□□ (secret key) □□□□□□□□□□□□□□□□ □□□□□□□□□□□□□ Section 15.3 □□□□□□□

1.4.1. □□□□□□□□□□ Docker Image

Python, Node.js, AWS CDK □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□/□□□□□□□□□□□□□□□□□ Docker image □□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ Docker □□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□

Chapter 2. □□□□□□

2.1. □□□□□□□

□□□□□□□□□□□ □□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□NIST(□□□□□□□□□□□□□□) □□□ The NIST Definition of Cloud Computing □□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ Figure 2 □□□□

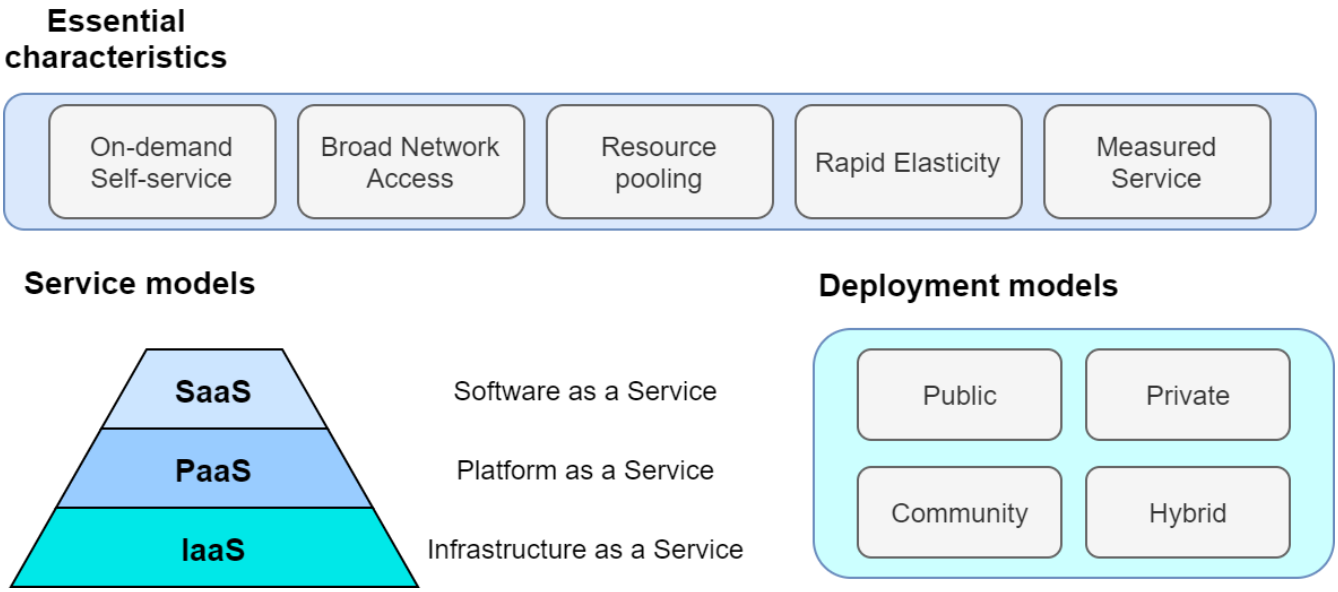


Figure 2. The NIST Definition of Cloud Computing

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□/□□□□□□□□□□□□□□□□□□□

- **On-demand self-service** □□□□□□□□□□□□□□□□□□□□

□□□

□□□

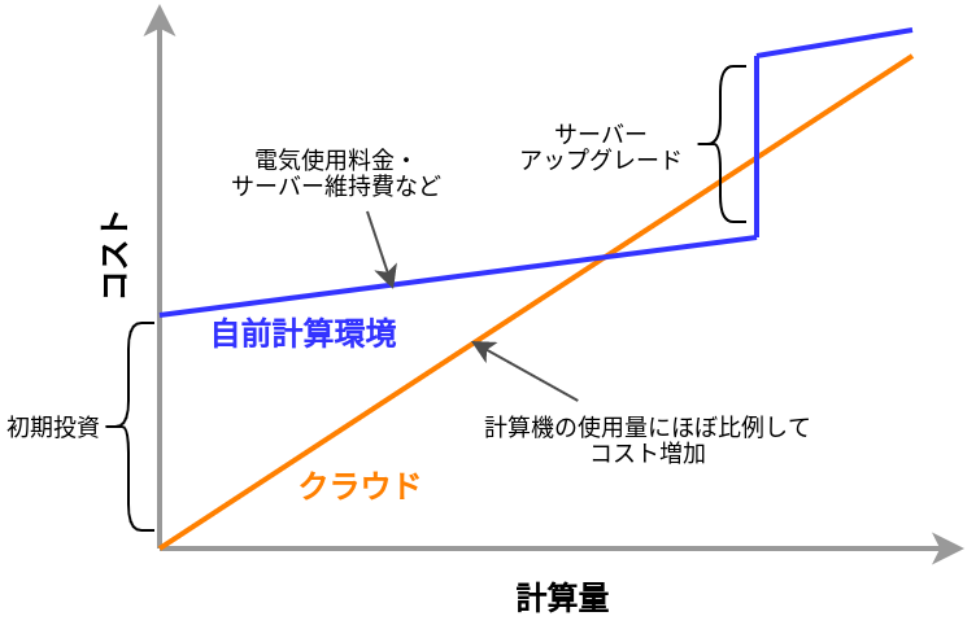
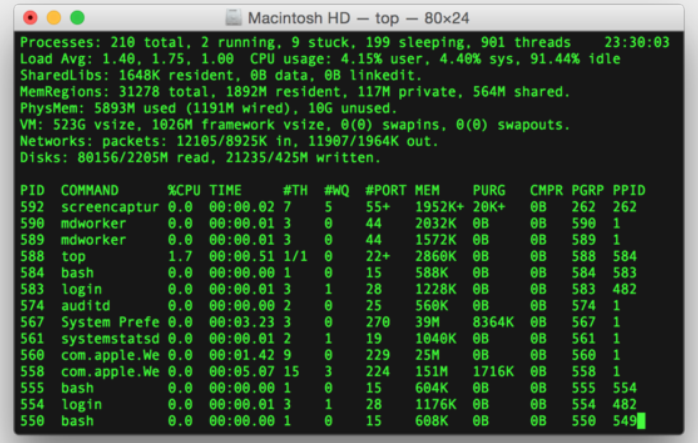

コスト
電気使用料金・
サーバー維持費など
サーバー
アップグレード
自前計算環境
初期投資
計算機の使用量にほぼ比例して
コスト増加
クラウド
計算量

Figure 3. □□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□**1.**□□□□□□□□□□□□□□□□□□□


Macintosh HD — top — 80×24
Processes: 210 total, 2 running, 9 stuck, 199 sleeping, 901 threads 23:30:03
Load Avg: 1.40, 1.75, 1.00 CPU usage: 4.15% user, 4.40% sys, 91.44% idle
SharedLibs: 1648K resident, 0B data, 0B linkedit.
MemRegions: 31278 total, 1892M resident, 117M private, 564M shared.
PhysMem: 5893M used (1191M wired), 10G unused.
VM: 523G vsize, 1026M framework vsize, 0(0) swapins, 0(0) swapouts.
Networks: packets: 12105/8925K in, 11907/1964K out.
Disks: 80156/2205M read, 21235/425M written.
PID COMMAND %CPU TIME #TH #WQ #PORT MEM PURG CMPR PGRP PPID
592 screencaptur 0.0 00:00.02 7 5 55+ 1952K+ 20K+ 0B 262 262
590 mdworker 0.0 00:00.01 3 0 44 2032K 0B 0B 590 1
589 mdworker 0.0 00:00.01 3 0 44 1572K 0B 0B 589 1
588 top 1.7 00:00.51 1/1 0 22+ 2860K 0B 0B 588 584
584 bash 0.0 00:00.00 1 0 15 588K 0B 0B 584 583
583 login 0.0 00:00.01 3 1 28 1228K 0B 0B 583 482
574 auditd 0.0 00:00.00 2 0 25 560K 0B 0B 574 1
567 System Prefe 0.0 00:03.23 3 0 270 39M 8364K 0B 567 1
561 systemstatsd 0.0 00:00.01 2 1 19 1040K 0B 0B 561 1
560 com.apple.We 0.0 00:01.42 9 0 229 25M 0B 0B 560 1
558 com.apple.We 0.0 00:05.07 15 3 224 151M 1716K 0B 558 1
555 bash 0.0 00:00.00 1 0 15 604K 0B 0B 555 554
554 login 0.0 00:00.01 3 1 28 1176K 0B 0B 554 482
550 bash 0.0 00:00.00 1 0 15 608K 0B 0B 550 549

□□□

Chapter 3. AWS□□

3.1. AWS□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ AWS □□□□□ □□□□□□□□□□□□□□□ AWS □□□□□□□□□□□□□□□□

AWS (Amazon Web Services) □Amazon□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ AWS □ Amazon □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□2006□□□□□□□□ 2021□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ (□32%) □□□□□□□□ (□□)□ Netflix □ Slack □□□□□□□□□□□□□□□□□□□□

□□□

3.3. Region □ Availability Zone

AWS □□□□

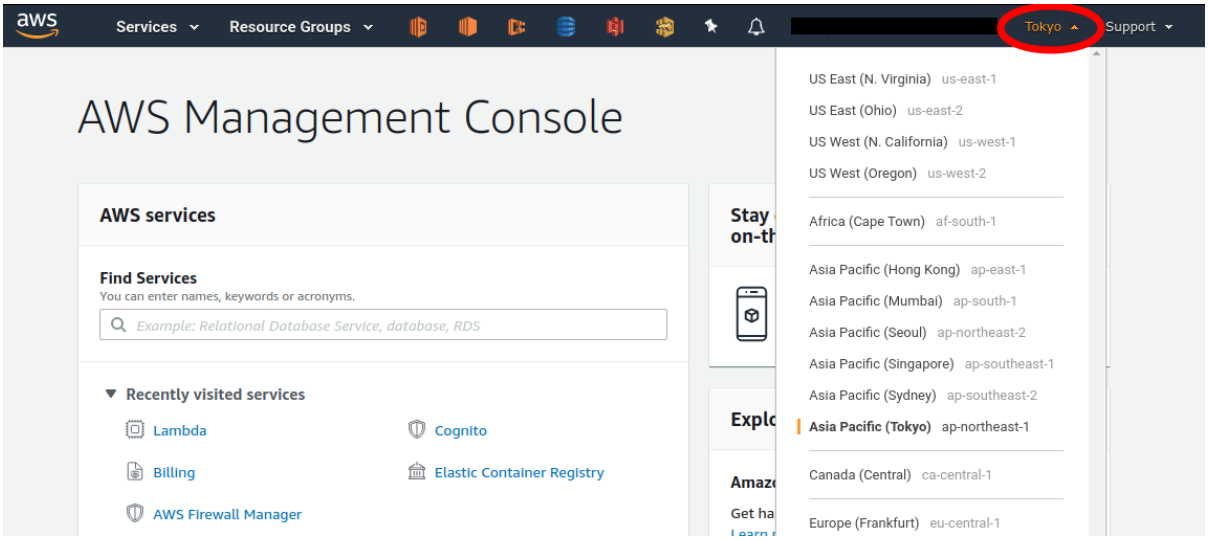


Figure 7. AWS□□□□□□□□□□□□□□□□

`Avaialibity Zone (AZ)` □□□□□□□□□□□□□□□□□

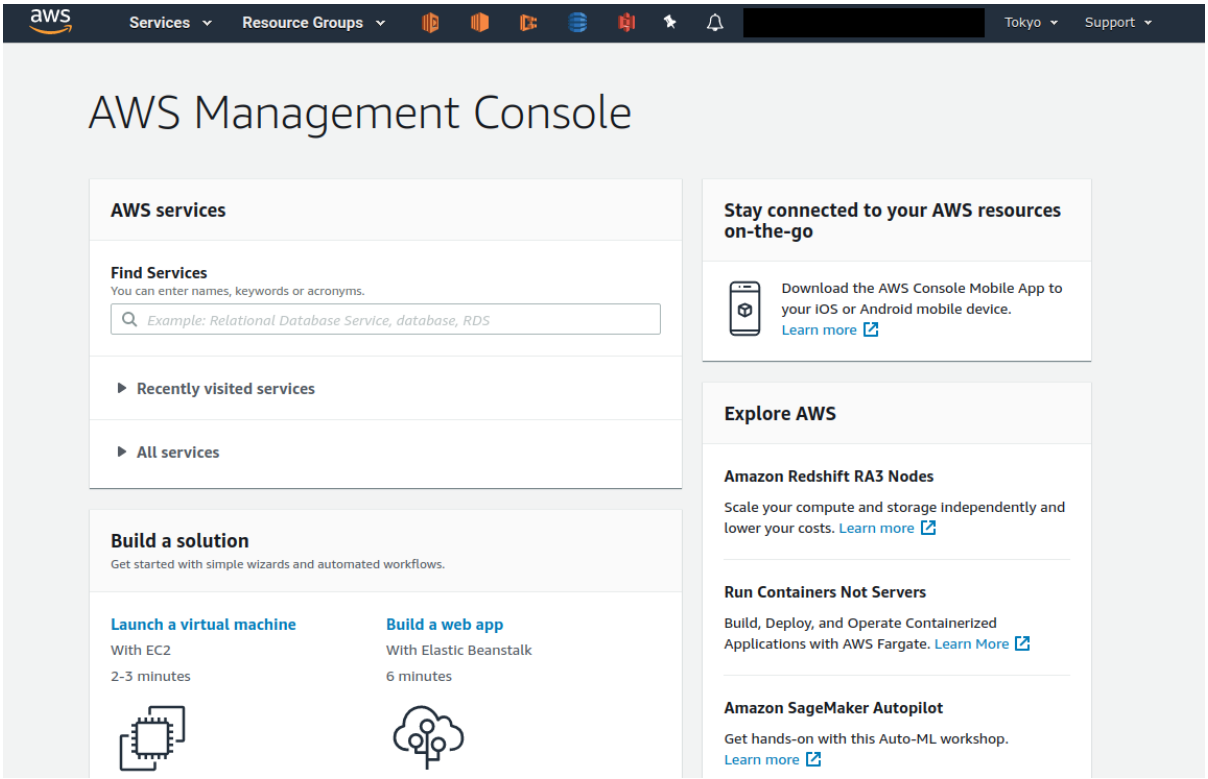


Figure 8. AWS□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□ EC2 □□□□□□□□□□□□□□□□□S3□□□□□□□□□□□□□□□□□□□□□□□□□□□□□AWS□□□□□□□□□□□□□□□ GUI (Graphical User Interface) □□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□

Python□□□□□□□□□□□□□□□□□□□□ boto3 □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

```
import boto3

s3_client = boto3.client("s3", region_name="ap-northeast-1")
s3_client.create_bucket(Bucket="my-bucket")
```

□□□□□□□□□□□□□

□□□EC2□□□□□□□□(□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□AWS□□□□□□□□□□□□□□□□

□□□□□□□□□□□□

□□□□□□□□AWS CLI □□□□ S3 □□□□□□□□□□□□□□□□□□□□□□□□ EC2 □ Lambda, DynamoDB □□□□□□□□□□□ AWS CLI □□□□□□□□□□□□□□□□□□□□

> **Amazon Resource Name (ARN)**
>
> AWS □□□□□□□□□□□□□□ Amazon Resource Name (ARN) □□□□□□ ID □□□□□□□□□□ ARN □ `arn:aws:s3:::my_bucket/` □□□□□□□□□□□□□□□□□ARN □□□□□□□□□□□□□□ AWS □□□□ (S3 □□□□□□ EC2 □□□□□□□) □□□□□□□□□□□□□□□□□□
>
> S3 □□□□□ EC2 □□□□□□□□□□□ ARN □□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□(□□□)□□□□□□□□□□□□□□□□

3.5.2. AWS CDK

□□□□□□□ CloudFormation □□□□□□□□

Further reading

- AWS CDK Examples: CDK□□□

Chapter 4. Hands-on #1: □□□□EC2□□□□□□□□□□□□

□□□□□□□□□□□□□ CDK □□□□ EC2 □□□□□□□□(□□□□□□□)□□□□□SSH□□□□□□□□□□□□□

```
$ ssh <user name>@<host name>
```

SSH □□□□

```
2
3     def __init__(self, scope: Construct, construct_id: str, key_name: str, **
  kwargs) -> None:
4         super().__init__(scope, construct_id, **kwargs)
5
6         ①
7         vpc = ec2.Vpc(
8             self, "MyFirstEc2-Vpc",
9             max_azs=1,
10            ip_addresses=ec2.IpAddresses.cidr("10.10.0.0/23"),
11            subnet_configuration=[
12                ec2.SubnetConfiguration(
13                    name="public",
14                    subnet_type=ec2.SubnetType.PUBLIC,
15                )
16            ],
17            nat_gateways=0,
18        )
19
20        ②
21        sg = ec2.SecurityGroup(
22            self, "MyFirstEc2Vpc-Sg",
23            vpc=vpc,
24            allow_all_outbound=True,
25        )
26        sg.add_ingress_rule(
27            peer=ec2.Peer.any_ipv4(),
28            connection=ec2.Port.tcp(22),
29        )
30
31        ③
32        host = ec2.Instance(
33            self, "MyFirstEc2Instance",
34            instance_type=ec2.InstanceType("t2.micro"),
35            machine_image=ec2.MachineImage.latest_amazon_linux(),
36            vpc=vpc,
37            vpc_subnets=ec2.SubnetSelection(subnet_type=ec2.SubnetType.PUBLIC),
38            security_group=sg,
39            key_name=key_name
40        )
```

① □□□□□□VPC□□□□□□

② □□□ security group (SG) □□□□□□□□ □□□□□□□□IPv4□□□□□□□□□□□□22 (SSH□□□□□□□□□□)□□□□□□□□□□□□□□□□□□□□□□□□□□

③ □□□□□□□□□□□ VPC□ SG □□□□□□ EC2 □□□□□□□□□□□□□□□ □□□□□□□□□□□ t2.micro □□□□□ Amazon Linux □OS□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□

4.3.1. VPC (Virtual Private Cloud)

VPC □□□□□

VPC □ AWS □□□□□

4.3.2. Security Group

Security group (SG) □□ EC2 □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□ IP □□□□□□□□□□□□□□□□□□□□□□□(□□□□□□□□□□□□□□□□□□) □□□□□□IP□□□□□□□□□□□□□□□□□□□ (□□□□□□□□□□□□□□□□□□□□□) □□□□□□□□□□

□□□□□□□□□□□□□□□□□

```
sg = ec2.SecurityGroup(
    self, "MyFirstEc2Vpc-Sg",
    vpc=vpc,
    allow_all_outbound=True,
)
sg.add_ingress_rule(
    peer=ec2.Peer.any_ipv4(),
    connection=ec2.Port.tcp(22),
)
```

□□□□□□□□□□ SSH □□□□□□□□□□□□□□□□□□□□ `sg.add_ingress_rule(peer=ec2.Peer.any_ipv4(), connection=ec2.Port.tcp(22))` □□□□□□□□□ IPv4 □□□□□□□□□□□22□□□□□□□□□□□□□□□□□□ □□□ SSH □ EC2 □□□□□□□□□□□□□□□□□□□□

Instance	vCPU	Memory (GiB)	Network bandwidth (Gbps)	Price per hour ($)
t2.medium	2	4	-	0.0464
c5.24xlarge	96	192	25	4.08
c5n.18xlarge	72	192	100	3.888
x1e.16xlarge	64	1952	10	13.344

Table 2 □□□□□□□□□ CPU □1□□□□96□□□□□□□□□□ 1GB □□ 2TB □□□□□□□□□□□□□□□□□□□100Gbps□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□CPU□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ EC2 □□□□□□□□□□□□□□□□□□□□

4.4.1. Python □□□□□□□□□□□□□□□□

□□□□Python □□□□□□□□□□□□□□□□□□□□□□□□□□□Python □□□□□□□□□□□□□□□□□□□□ venv □□□□□□

□□□□ `handson/ec2-get-started` □□□□□□□□□□□□□□□□

```
$ cd handson/ec2-get-started
```

□□□□□□□□□□□□□□□ `venv` □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

```
$ python3 -m venv .env
$ source .env/bin/activate
$ pip install -r requirements.txt
```

□□□ Python □□□□□□□□□□□

> `venv` □□□□□□□□ Section 15.7 □□□□□□□□□

> □□□□□□□□ `pip` □□□□ `pip3` □□□□ `python3 -m pip` □□□□□□□□□□□□□□

4.4.2. AWS □□□□□□□□□□□□□□□□

AWS CLI □□□ AWS CDK □□□□□□ AWS □□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□ Section 15.2 □□□□□□□□ □□□□□□□□□□□□□□□□□□ Section 15.3 □□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□ `AWS_ACCESS_KEY_ID` □□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□ `~/.aws/credentials` □□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□ AWS CLI/CDK □□□□□□□□□□□□□□□□□□□□

4.4.4. □□□□□□□

□□□□□□□□□□□□ EC2 □□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□□□ AWS □□□□□□□□□□□ -c key_name="HirakeGoma" □□□□□□□□□□□□□□□□□□□□□ HirakeGoma □□□□□□□□□□□□□□□□□□□□□□□□□□

```
$ cdk deploy -c key_name="HirakeGoma"
```

□□□□□□□□□□□□□□□ VPC□ EC2 □□□AWS□□□□□□□□□ □□□□□□□□□□□□□□□□□□ Figure 11 □□□□□□□□□□□□□□□□□□□□ □□□□□ InstancePublicIp □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ **IP** □□□□□□□□□ IP □□□□□□□□□□□□□□□□□□□□

```
$ cat /proc/cpuinfo

processor   : 0
vendor_id   : GenuineIntel
cpu family  : 6
model       : 63
model name  : Intel(R) Xeon(R) CPU E5-2676 v3 @ 2.40GHz
stepping    : 2
microcode   : 0x43
cpu MHz     : 2400.096
cache size  : 30720 KB
```

□□□□□□□□□□□□□□□□□□□□□□□□□□□□

```
$  top -n 1

top - 09:29:19 up 43 min,  1 user,  load average: 0.00, 0.00, 0.00
Tasks:  76 total,   1 running,  51 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.3%us,  0.3%sy,  0.1%ni, 98.9%id,  0.2%wa,  0.0%hi,  0.0%si,  0.2%st
Mem:   1009140k total,   270760k used,   738380k free,    14340k buffers
Swap:        0k total,        0k used,        0k free,   185856k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
    1 root      20   0 19696 2596 2268 S  0.0  0.3   0:01.21 init
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kthreadd
    3 root      20   0     0    0    0 I  0.0  0.0   0:00.00 kworker/0:0
```

`t2.micro` □□□□□□□□□□□ 1009140k = 1GB □□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□ Python 2 □□□□□□□□□□□□ Python 3 □□□□□□□□ Python 3.6 □□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□

```
$ sudo yum update -y
$ sudo yum install -y python36
```

□□□□□□□□ Python □□□□□□□□□

```
$ python3
Python 3.6.10 (default, Feb 10 2020, 19:55:14)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-28)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Python □□□□□□□□□□□□□□□ `Ctrl + D` □□□□ `exit()` □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ (□□□□□□□□□□□□□□□□□□□□□□□□□) □ □□□□□□□□□□□□□□□□□

```
$ exit
```

4.4.7. AWS □□□□□□□□□□

□□□□□□□□□□□□□□□□□□□ EC2 □□□□□□□□□□□□□□□ EC2□□AWS □□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□

□□□□□□□□□□□□□□□□ AWS □□□□□□□□□□□□□□□□ □□□□□□□□□ Services □□ EC2 □□□(□□)□□□ □□□□□□□□□□□□ Instances □□□□□□□□□. □□□□ Figure 13 □□□□□□□□□□□□□□□□□□□□ □□ □□□□VPC□SG □□□□□□□□□□□□□□□□□□□□□□

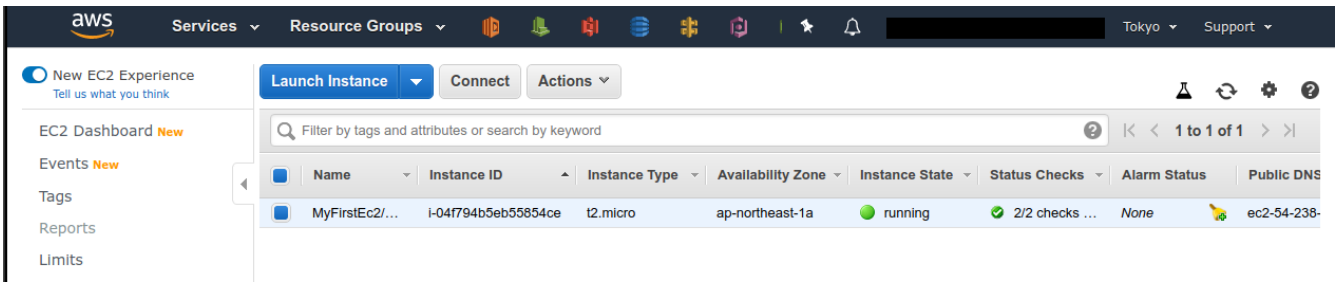


Figure 13. EC2 □□□□□□□□

> □□□□□□□□□□□□□□□□□□□ **(□□□□□□□ ap-northeast-1, Tokyo)** □□□□□□□□□□□□□□□□□

□□□ CloudFormation □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□CloudFormation □□□□□□□□□□□□□□□□□ □□□□ **(stack)** □□□ AWS □□□□□□□□□□□□□□□□ □□□□□□□□ VPC/EC2/SG □□□□□□□□□□□□□□□□□□□ □□□□□□□ CloudFormation □□□□□□□□□□□□ (Figure 14)□

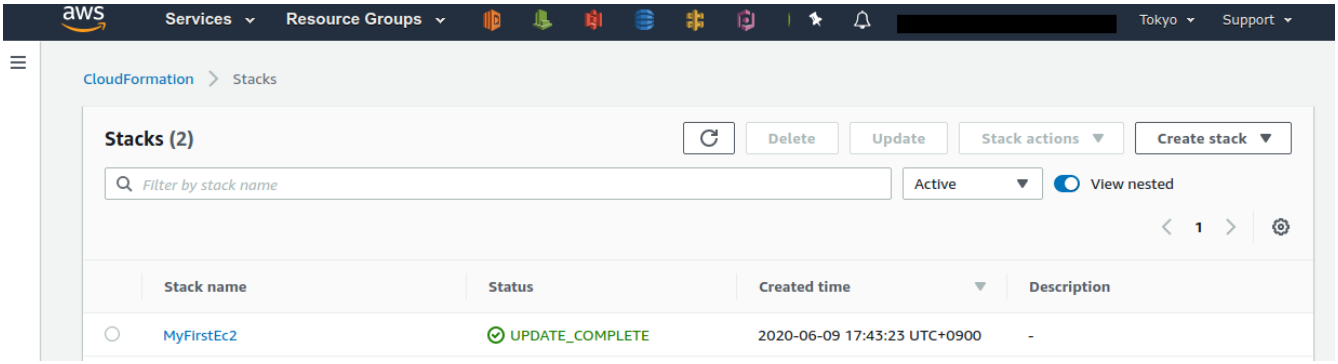


Figure 14. CloudFormation □□□□□□□□

"MyFirstEc2" □□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□EC2, VPC □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

4.4.8. □□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□ Cloudformation □□□□□□□□□□□□ "Delete" □□□□□□□□□□□□□ (Figure 15)□ □□□□□□□□□□□□□□□ "DELETE_IN_PROGRESS" □□□□□□□□□□□□□□□ CloudFormation □□□□□□□□□□□□□□□□□□

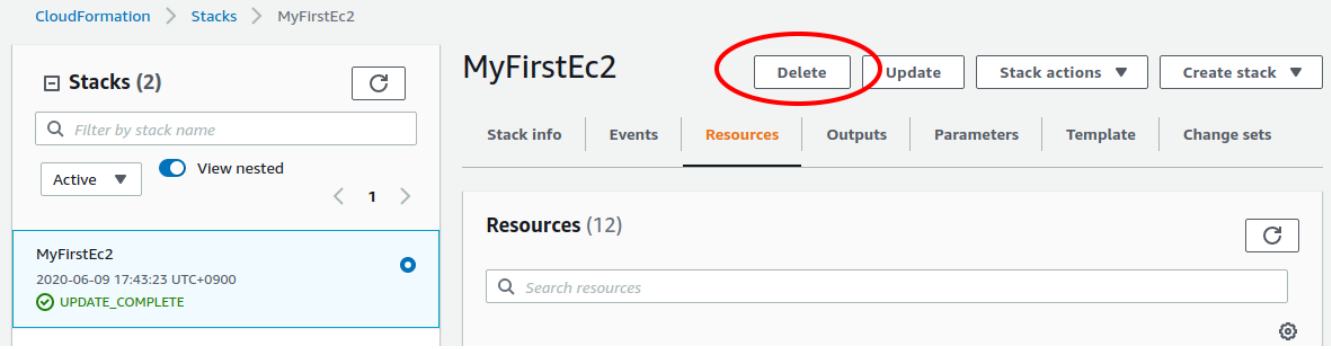


Figure 15. CloudFormation□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□

```
$ cdk destroy
```

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□VPC, EC2 □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ CloudFormation □□□□□□□□□□□□□□□□□□ AWS □□

□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□ EC2 □□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□ SSH □□□□□□□□□□□□□□□□□□□□□ □□□ EC2 □□□□□□□□□□□□□□□□□□□□

□□□□□□ EC2 □□□□□□□□□□□□□□□□□□□□□□□□□□ SSH □□□□□□□□□□□□□

4.5. □□

□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Chapter 2 □□□□□□□□□□□□□□□□□□□□

Chapter 5. □□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□

*Figure 17. GPU□□□□□□□□□□GPU□□□□□□□□□□

□□□□□□□□□□ V100 □□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□

- □□□□□□ Deep Learning (□□ □□) □□□□□□□□□□□□□ □□□□□□□□□□□□□

Chapter 6. Hands-on #2: AWS □□□□□□□□□□□□□□

6.1. □□

□□□□□□□□□□□□ GPU □□□□□EC2□□□□□□□□□□□□□

- GPU□□□□□ g4dn.xlarge □□□□□□□□□□□□
- □□□□□□□□□□□□

Chapter 5 □□□□□□□□□□□ NVIDIA T4 □□□□□□□□□□□□□ GPU □□□□□□□□□□□□□□□□ CPU □ 4 core, □□□□□□□□ 16GB □□□□□□□□□□□□□

② □□□□□Deep Learning □□□□□□□□□□□□□□□□□□□□□□□□□AMI (Deep Learning Amazon Machine Image; DLAMI) □□□□□□□ (□□□□□□Amazon Linux □□□AMI□□□□□□□□)□ □□□□ AMI □ID□ □□□□□□□□□□□□□□□□□□□□□□□□□□□□ us-east-1 □ ap-northeast-1 □□□□□□□□□□□□□

DLAMI □□□□□□□□□□□□□□□□□□□□□□□□□

> AMI □ us-east-1 □ ap-northeast-1 □□□□□□□□□□□□□□□□□□□□

```
tomoyuki@balthasar:02-ec2-dnn$ aws ec2 describe-images --owners amazon --image-ids "ami-09c0c16fc46a29ed9"
{
    "Images": [
        {
            "Architecture": "x86_64",
            "CreationDate": "2020-05-20T14:47:04.000Z",
            "ImageId": "ami-09c0c16fc46a29ed9",
            "ImageLocation": "amazon/Deep Learning AMI (Amazon Linux 2) Version 29.0",
            "ImageType": "machine",
            "Public": true,
            "OwnerId": "898082745236",
            "PlatformDetails": "Linux/UNIX",
            "UsageOperation": "RunInstances",
            "State": "available",
            "BlockDeviceMappings": [
                {
                    "DeviceName": "/dev/xvda",
                    "Ebs": {
                        "DeleteOnTermination": true,
                        "SnapshotId": "snap-0bd381ab76e5a6146",
                        "VolumeSize": 90,
                        "VolumeType": "gp2",
                        "Encrypted": false
                    }
                }
            ],
            "Description": "MXNet-1.6.0, Tensorflow-2.1.0 & 1.15.2, PyTorch-1.4.0 & 1.5.0, Neuron, & other f
rameworks, NVIDIA CUDA, cuDNN, NCCL, Intel MKL-DNN, Docker, NVIDIA-Docker & EFA support. For fully managed e
xperience, check: https://aws.amazon.com/sagemaker",
            "EnaSupport": true,
            "Hypervisor": "xen",
            "ImageOwnerAlias": "amazon",
            "Name": "Deep Learning AMI (Amazon Linux 2) Version 29.0",
            "RootDeviceName": "/dev/xvda",
            "RootDeviceType": "ebs",
            "SriovNetSupport": "simple",
            "VirtualizationType": "hvm"
        }
    ]
}
```

Figure 20. AMI ID = ami-09c0c16fc46a29ed9 □□□□□

Figure 20 □□□□□

□□□□□□□□□□□□□□□1□□□□□□□□□□□□ □□□□□□□□□□□□□□□□ (# □□□□□□□□□□□□□□□)□ □□□□□□□□□□□□□□□□

```
ssh -L localhost:8931:localhost:8888 ec2-user@0.0.0.0
```

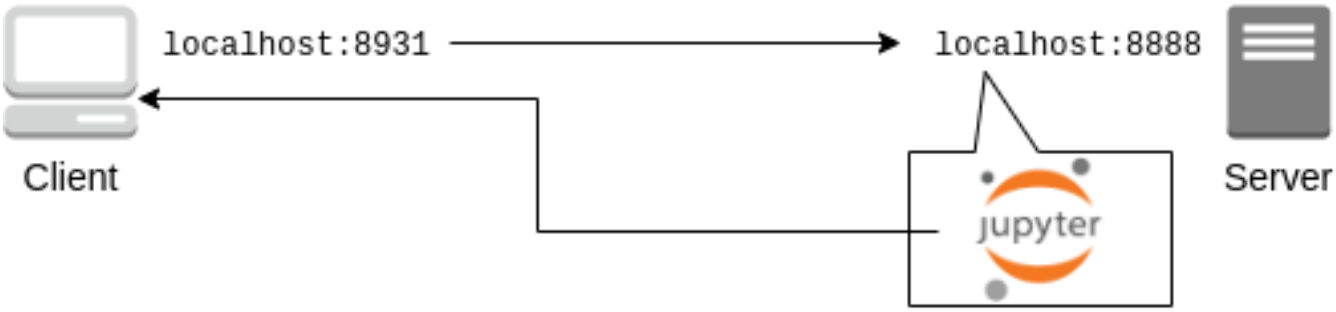


Figure 22. SSH □□□□□□□□□□□□□□□□ Jupyter Notebook □□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ (`:8931`, `:8888` □□) □□1□□65535□□□□□□□□□□□□□□□□ □□□□□□□□□ □□□ 22 (SSH) □□□□ 80 (HTTP) □□ □□□ Jupyter Notebook □□□□□□□□□□□□8888□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□8888□□□□□□□□□

SSH □□□□□□□□□□ `<IP address>` □□□□□□□□□□□□□□□IP□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□ **Docker** □□□□□□□□□□□□□□□□□

SSH □□□□□□□□□□ **Docker** □□ (□□□□□□□□□□□□□□□□□□) □□□□□□□□□□□□□□ □□□□□Jupyter □□□□□□□□□□□□ Docker □□□□□□□□□□□□

□□□□□□□□□ Docker □□□□□□□□□□□□□□□□□□ □□□□□□□□□□□ `cat ~/.ssh/HirakeGoma` □□□ □□□□ `-v` □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ (□□□□ Docker □□□□□□□□□□□□□□ "Use volumes" □□□)□

SSH□□□□□□□□□□□□□□□□□□□ GPU □□□□□□□□□□□□□□ □□□□□□□□□□□□□□

```
$ nvidia-smi
```

Figure 23 □□□□□□□□□□□□□□□□□□□□ □□□□□□□□ Tesla T4 □□GPU□1□□□□□□□□□□□□□□□□□□□□ □□□□GPU Driver □ CUDA □□□□□□□□ GPU □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

```
[ec2-user@ip-10-10-1-172 ~]$ nvidia-smi
Sat Jun 13 04:55:22 2020
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 440.33.01    Driver Version: 440.33.01    CUDA Version: 10.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla T4            On   | 00000000:00:1E.0 Off |                    0 |
| N/A   31C    P8    11W /  70W |      0MiB / 15109MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Figure 23. nvidia-smi □□□

6.5. Jupyter Notebook □□□

Jupyter Notebook □□□□□□□□□□□□□ Python □□□□□□□□□□□□□

```
$ cd ~ # go to home directory
$ jupyter notebook
```

□□□□□□□□□□□□□□□ Figure 25 □□□□□□□□□□□□□□□□□□□ □□□□□□□□Jupyter □□□□□□□ EC2 □□□□□□□□ localhost:8888 □□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□ localhost:8888 □□□ ?token=XXXX □□□□□□□□□□□□□□□□□□□□□□□□□□□□□

```
[I 06:01:10.466 NotebookApp] The Jupyter Notebook is running at:
[I 06:01:10.466 NotebookApp] http://localhost:8888/?token=537fee42934a7db9d0540024260d305b08bb0bf9fc41c5a7
[I 06:01:10.466 NotebookApp]  or http://127.0.0.1:8888/?token=537fee42934a7db9d0540024260d305b08bb0bf9fc41c5a7
[I 06:01:10.466 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[W 06:01:10.469 NotebookApp] No web browser found: could not locate runnable browser.
[C 06:01:10.469 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///home/ec2-user/.local/share/jupyter/runtime/nbserver-11720-open.html
    Or copy and paste one of these URLs:
        http://localhost:8888/?token=537fee42934a7db9d0540024260d305b08bb0bf9fc41c5a7
     or http://127.0.0.1:8888/?token=537fee42934a7db9d0540024260d305b08bb0bf9fc41c5a7
```

Figure 25. Jupyter Notebook □□□□□□□□

> Jupyter Notebook □□□□□□□□□□□□□□□□□□□□

- `Shift` + `Enter`: □□□□□
- `Esc`: **Command mode** □□□
- □□□□□□□ "+" □□□ □□□ Command mode □ `A` ⇒ □□□□□
- □□□□□□□ "□□□" □□□ □□□ Command mode □ `X` ⇒ □□□□□

□□□□□□□□□□□□□ Ventsislav Yordanov □□□□□□□ □□□□□□□

6.6. PyTorch□□□□□□

PyTorch □ Facebook AI Research LAB (FAIR) □□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ (Figure 28)□

```
In [1]: import torch

        print("Is CUDA ready?", torch.cuda.is_available())

        Is CUDA ready? True

In [3]: # create a random array in CPU
        x = torch.rand(3,3)
        print(x)

        tensor([[0.6896, 0.2428, 0.3269],
                [0.0533, 0.3594, 0.9499],
                [0.9764, 0.5881, 0.0203]])

In [4]: # create another array in GPU device
        y = torch.ones_like(x, device="cuda")
        # move 'x' from CPU to GPU
        x = x.to("cuda")

In [5]: # run addition operation in GPU
        z = x + y
        print(z)

        tensor([[1.6896, 1.2428, 1.3269],
                [1.0533, 1.3594, 1.9499],
                [1.9764, 1.5881, 1.0203]], device='cuda:0')

In [6]: # move z from GPU to CPU
        z = z.to("cpu")
        print(z)

        tensor([[1.6896, 1.2428, 1.3269],
                [1.0533, 1.3594, 1.9499],
                [1.9764, 1.5881, 1.0203]])
```

Figure 28. PyTorch□□□□□

□□□□ PyTorch □□□□□□□□□□□□□□ GPU □□□□□□□□□□□□□□□□□□□

```
import torch
print("Is CUDA ready?", torch.cuda.is_available())
```

□□:

```
Is CUDA ready? True
```

□□□3x3 □□□□□□□□□□ **CPU** □□□□□□□□□□

```
x = torch.rand(3,3)
print(x)
```

□□:

```
tensor([[0.6896, 0.2428, 0.3269],
        [0.0533, 0.3594, 0.9499],
        [0.9764, 0.5881, 0.0203]])
```

□□□□□□ **GPU** □□□□□□□□

```
y = torch.ones_like(x, device="cuda")
x = x.to("cuda")
```

□□□□□□□ x □ y □□□□□**GPU**□□□□□□□□

```
z = x + y
print(z)
```

□□:

```
tensor([[1.6896, 1.2428, 1.3269],
        [1.0533, 1.3594, 1.9499],
        [1.9764, 1.5881, 1.0203]], device='cuda:0')
```

□□□□ GPU □□□□□□□□□ CPU □□□□

```
z = z.to("cpu")
print(z)
```

□□:

```
tensor([[1.6896, 1.2428, 1.3269],
        [1.0533, 1.3594, 1.9499],
        [1.9764, 1.5881, 1.0203]])
```

□□□□□□□ GPU □□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□5.8□□□□□□□□□□□□□□ (□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□)□

```
CPU times: user 11.5 s, sys: 140 ms, total: 11.6 s
Wall time: 5.8 s
```

□□□□ GPU □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

```
s = 10000
device = "cuda"
x = torch.rand(s, s, device=device, dtype=torch.float32)
y = torch.rand(s, s, device=device, dtype=torch.float32)
torch.cuda.synchronize()

%time z = torch.matmul(x,y); torch.cuda.synchronize()
```

□□□□□□□□□□□□□□□□□□□□□□□ GPU □□ 553□□□ □□□□□□□□□□□□□□□□□

```
CPU times: user 334 ms, sys: 220 ms, total: 554 ms
Wall time: 553 ms
```

PyTorch □□□□□□ GPU □□□□□□ asynchronous (□□□) □□□□□□□□ □□□□□□□□□□□□□□□□□□□□

Figure 29. MNIST □□□□□□□□□□□□

□□□□ MNIST □□□□□□□□□□□□□□□□□□□□□□□□□□□□ **(Convolutional Neural Network; CNN)** □□□□□□□□ □□□□□□□□ /handson/minist/pytorch/ □□□ `mnist.ipynb` □ `simple_mnist.py` □□□□ □□□□□□□□□□□□□ PyTorch □□□ Example Project □ □□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□ `simple_mnist.py` □□□□□□□□□□□ (Figure 30)□ □□□□□ "Upload" □□ □□ Python □□□□□□□□□CNN □□□□□□□□□□□□□□□□□□□□

```python
import torch
import torch.optim as optim
import torchvision
from torchvision import datasets, transforms
from matplotlib import pyplot as plt

# custom functions and classes
from simple_mnist import Model, train, evaluate
```

torchvision □□□□□□□□MNIST □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ (Model, train, evaluate) □□□□□□□□□□□□□□□

□□□MNIST □□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

```python
transf = transforms.Compose([transforms.ToTensor(),
                             transforms.Normalize((0.1307,), (0.3081,))])

trainset = datasets.MNIST(root='./data', train=True, download=True, transform=transf)
trainloader = torch.utils.data.DataLoader(trainset, batch_size=64, shuffle=True)

testset = datasets.MNIST(root='./data', train=False, download=True, transform=transf)
testloader = torch.utils.data.DataLoader(testset, batch_size=1000, shuffle=True)
```

□□□□ MNIST □□□□ 28x28 □□□□□□□□□□□□(□□□□)□□□□□□□□□□□□(0 - 9 □□□)□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ Figure 31 □□□□□□□□□□□□□□□□□□□□

```python
examples = iter(testloader)
example_data, example_targets = examples.next()

print("Example data size:", example_data.shape)

fig = plt.figure(figsize=(10,4))
for i in range(10):
    plt.subplot(2,5,i+1)
    plt.tight_layout()
    plt.imshow(example_data[i][0], cmap='gray', interpolation='none')
    plt.title("Ground Truth: {}".format(example_targets[i]))
    plt.xticks([])
    plt.yticks([])
plt.show()
```

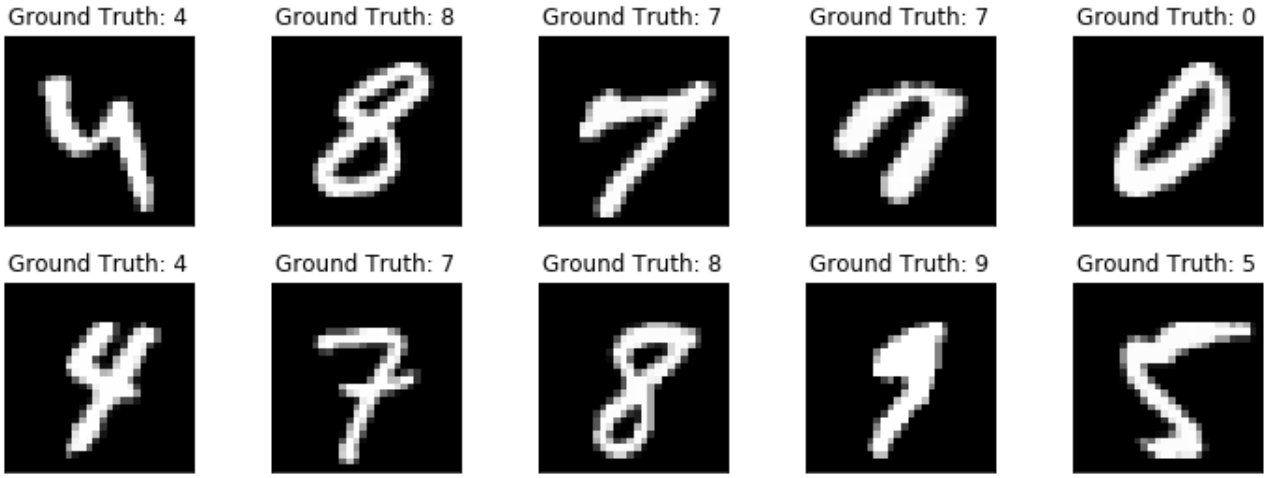


Figure 31. MNIST □□□□□□□□□□□□□□□□□□

□□□ CNN □□□□□□□□□□□

```
model = Model()
model.to("cuda") # load to GPU
```

□□□□ Model □ simple_mnist.py □□□□□□□□□□□ □□□□□□□Figure 32 □□□□□□□□□□□□□□□□□□2□□□□□□□□□□□□□□□□□

```
    print(f"\nTest set: Average loss: {test_loss:.4f}, Accuracy: {test_accuracy:.1f}%\n")

plt.figure(figsize=(7,5))
plt.plot(train_losses)
plt.xlabel("Iterations")
plt.ylabel("Train loss")
plt.show()
```

□□□□5□□□□□□□□□□□□□□□ GPU □□□□□□□□□□□□□□□□□□□□1□□□□□□□□□□□□□

□□□□□□ Figure 33 □□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□ (Loss function) □□□□□□□□□ (=□□□□□□□□□□) □□□□□□□□

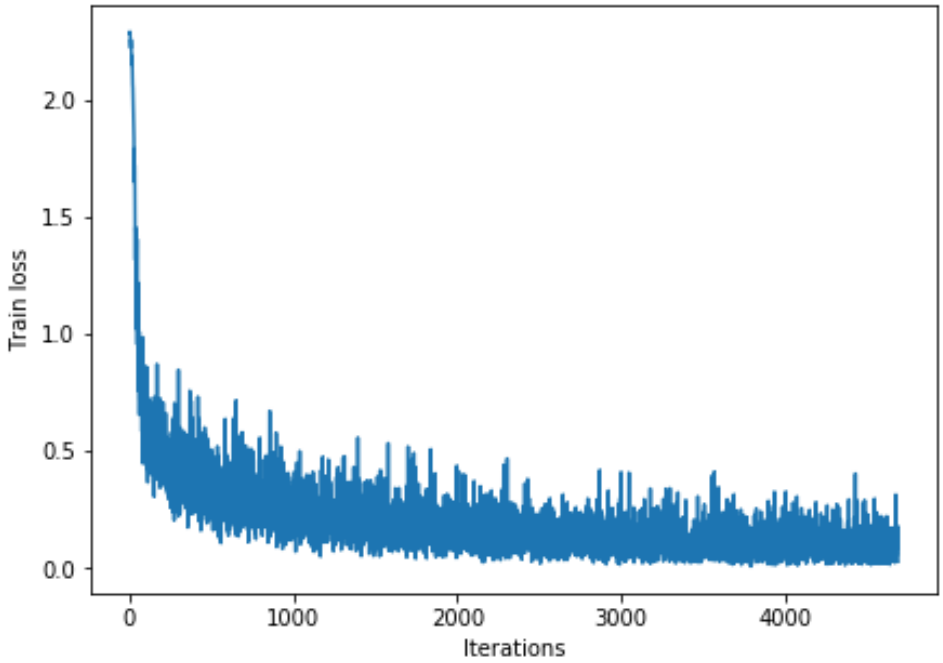


Figure 33. □□□□□□□□□□ Train loss □□□

□□□□□□□□□□□□□□□□□□□

```
    plt.tight_layout()
    plt.imshow(example_data[i][0], cmap='gray', interpolation='none')
    plt.title("Prediction: {}".format(output.data.max(1, keepdim=True)[1][i].
item()))
    plt.xticks([])
    plt.yticks([])
plt.show()
```

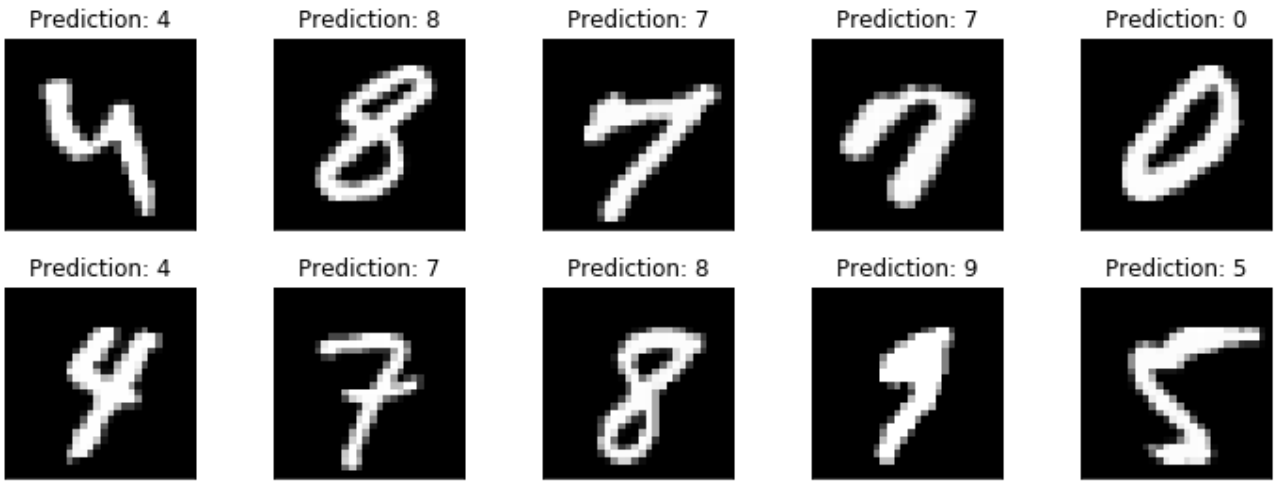


Figure 35. □□□□ *CNN* □□□□*MNIST*□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ mnist_cnn.pt □□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□ AWS Budgets □□□□□□□□□□□□□□□□□□□

Chapter 7. Docker □□

□□□

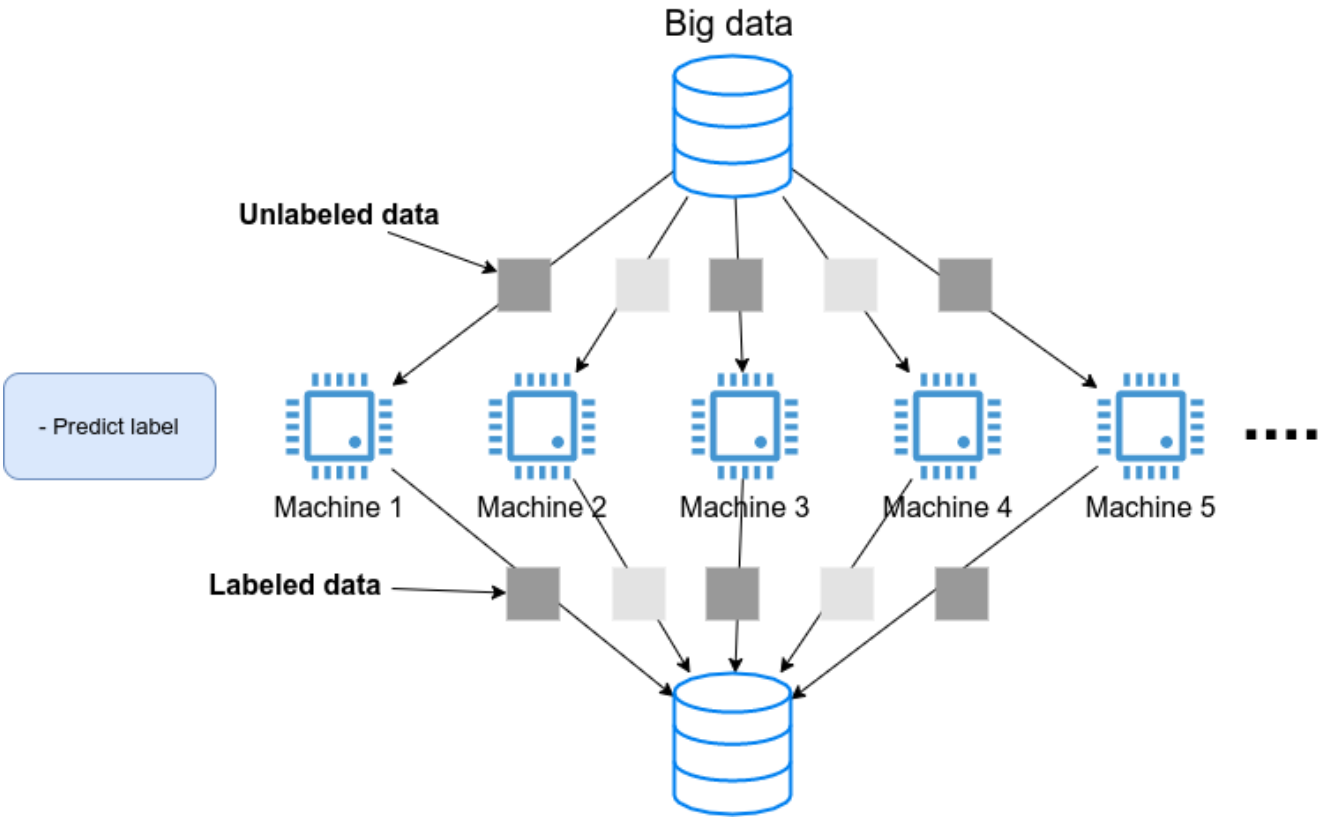

Big data
Unlabeled data
- Predict label
Machine 1
Machine 2
Machine 3
Machine 4
Machine 5
....
Labeled data

Figure 37. □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□

□□□□□ (VM) □□□□□□□□□□□□□□□□□□□□□□□□□□OS□□□□□□□□□□□ (Figure 39)□ VM □□ □□□□□□□□ **(Hypervisor)** □□□□□□□□□□□□□□□□ Hypervisor □□□□□□□□□□□□□□□□ (CPU, RAM, network □□) □□□□□□□□□□□ □□□□□ □□□□□□□□□□□□ CPU □4□□□□□□□□□□□□□□□□□□□□□□□ (2,2) □□□□□□□□□□□□□□□□□□□□□ VM □□□□□□ OS □□□ VM □□□□□□ OS □□□□□□□□□□□□□□□□□□□□□ OS-A □ OS-B □□□□□□□□□CPU□□□□□□□□□□□□□□□□□□□□□□□□□□ (□□□ isolation □□□)□ VM □□□□□□□□□□□□□□□□□□□□□□□□□□□□ VMware□ VirtualBox□ Xen □□□□□□ □□□□□□□□□□□□□ EC2 □□□□□□ VM □□□□□□□□□□□□□□□□□□□□

DevOps □ CI / CD (Continuous Integration / Continuous Deployment) □□□□□□□□□□□□□□□□ Docker □□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□ (Chapter 11) □□□□□□□□□□□□□□□□□□□□

7.3.3. □□□□□□□□

Pull □□□□□□□□□□□□□□□□□ `run` □□□□□□□□□

```
$ docker run -it ubuntu:18.04
```

□□□□ `-it` □□□□□□□□□□□□□ shell □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□ Ubuntu □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ (Figure 41)□ □□□□□□□□□□□□□□□□□□ (□□□□□) □□□□ **Container (□□□□)** □□□□

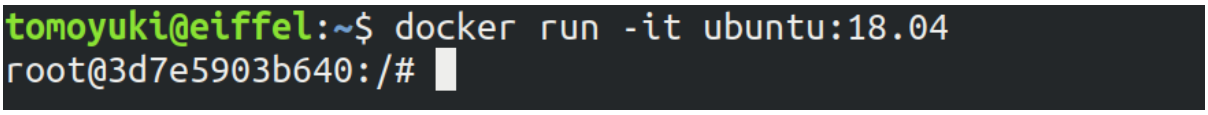


Figure 41. Docker □□□□ ubuntu:18.04 □□□□□□□□

□□□□□□□□ `ubuntu:18.04` □□□□□□□□□□ Ubuntu OS □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□ Chapter 6 □□□ DLAMI □□□□□□□□□□□□□ □□□□□ PyTorch □□□□□□□□□□□□□□□□□□ PyTorch □□□ Docker Hub □□□□□ □□□□□□□□□□□

□□□□□□□□□□□□□

```
$ docker run -it pytorch/pytorch
```

> `docker run` □□□□□□□□□□□□□□□□□□□□

```
FROM node:12
LABEL maintainer="Tomoyuki Mano"

RUN apt-get update \
    && apt-get install nano

①
RUN cd /opt \
    && curl -q "https://www.python.org/ftp/python/3.7.6/Python-3.7.6.tgz" -o Python-
3.7.6.tgz \
    && tar -xzf Python-3.7.6.tgz \
    && cd Python-3.7.6 \
    && ./configure --enable-optimizations \
    && make install

RUN cd /opt \
    && curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
"awscliv2.zip" \
    && unzip awscliv2.zip \
    && ./aws/install

②
RUN npm install -g aws-cdk@1.100

# clean up unnecessary files
RUN rm -rf /opt/*

# copy hands-on source code in /root/
COPY handson/ /root/handson
```

Dockerfile □□□□

podman □ Red Hat □□□□

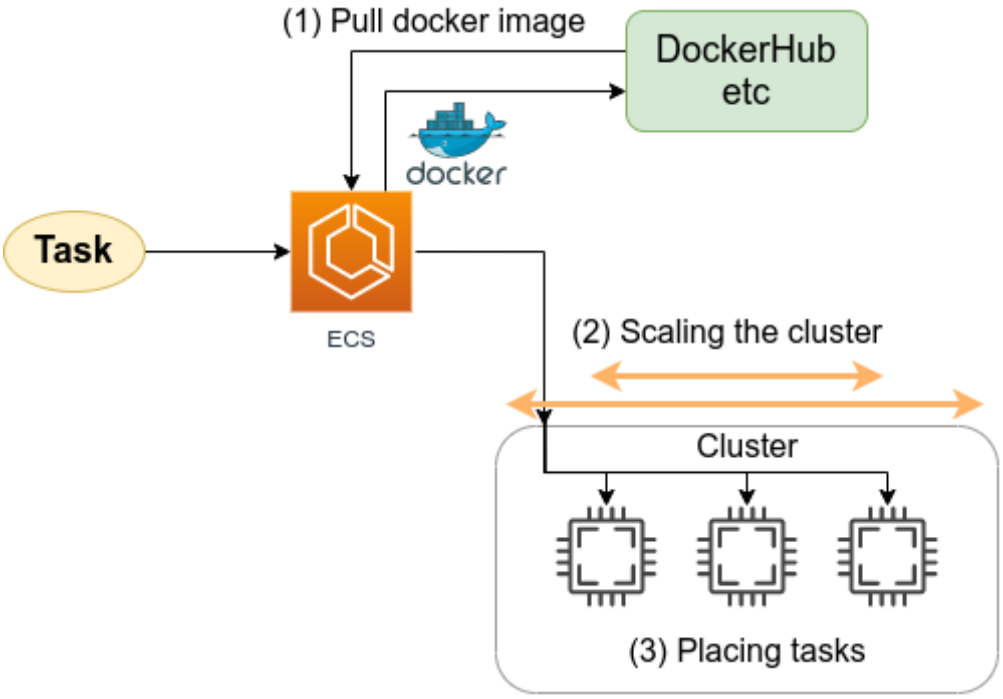


Figure 43. ECS □□□

Chapter 8. Hands-on #3: AWS □□□□□□□□□□□□□□□□

□□□□□□□□□□□□ Docker □ ECS □□□□□□□□□□□□□□□□

8.2. □□

□□□□□□□□□□□□□ GitHub □ handson/qa-bot □□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ (Section 4.1) □□□□□□□□□□□□□□□□ □□□ Docker □□□□□□□□□□□□□□□□□□□□

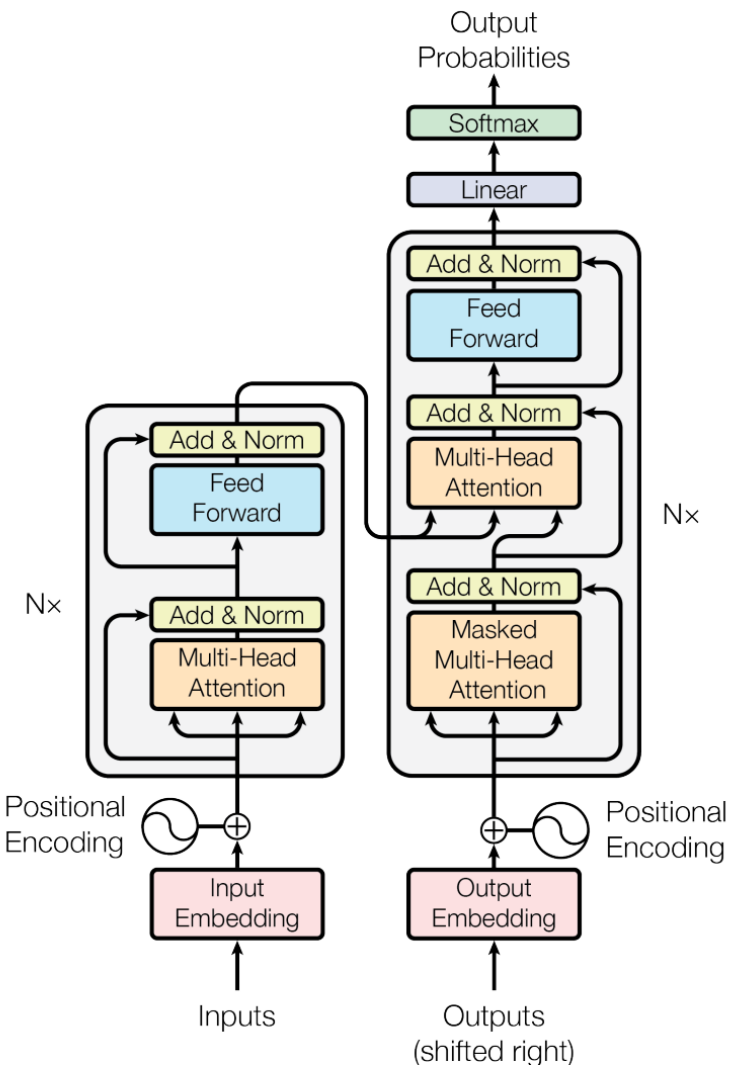


Figure 45. Transformer □□□□□□□□□□□ (□□□□: Vaswani+ 2017)

□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

```
$ docker run tomomano/qabot "${context}" "${question}" foo --no_save
```

□□□□□□□ Docker image □□□□□□□□ context □□□□□□□□□□□□□□□ question □□□□□□□□□□□□□□□□□□□□

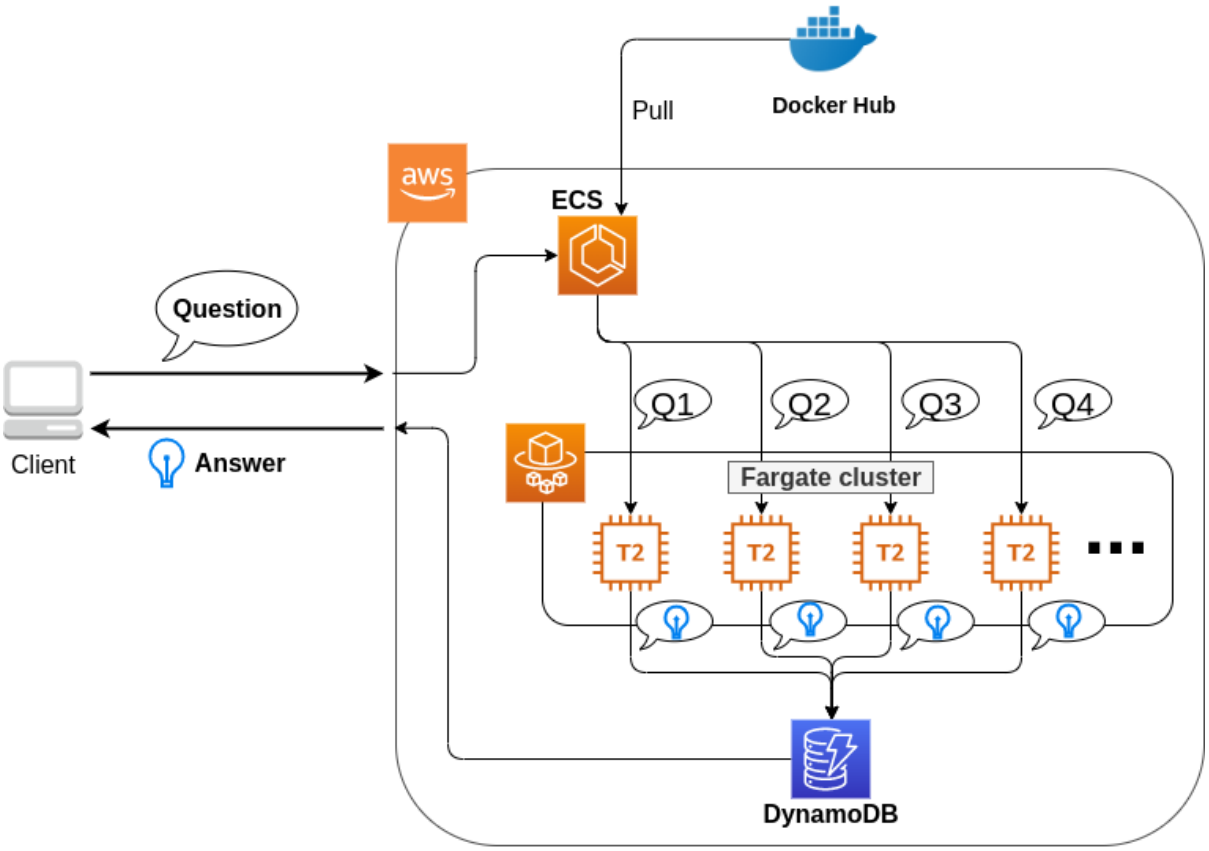


Figure 46. □□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□

- □□□□□□□□□□□□ AWS □□□□□□□□□□□□□□□□□□
- □□□□□□□□ ECS □□□□□□□□□□□□
- ECS □□ Docker Hub □□□□□□□□□□□□□□□□□□□
- □□□ECS □□□□□□□□□□□□ Fargate □□□□□□□□□□□□□□□□□□□□□□□□□□ Docker □□□□□□□□□□□□□□□□□□□□□□□□□□
  - □□□□□□□□□□□□□□□□□ Fargate □□□□□□□□□□□□□□□□□□□□

```
11                 name="item_id", type=dynamodb.AttributeType.STRING
12             ),
13             billing_mode=dynamodb.BillingMode.PAY_PER_REQUEST,
14             removal_policy=cdk.RemovalPolicy.DESTROY
15         )
16
17         ②
18         vpc = ec2.Vpc(
19             self, "EcsClusterQaBot-Vpc",
20             max_azs=1,
21         )
22
23         ③
24         cluster = ecs.Cluster(
25             self, "EcsClusterQaBot-Cluster",
26             vpc=vpc,
27         )
28
29         ④
30         taskdef = ecs.FargateTaskDefinition(
31             self, "EcsClusterQaBot-TaskDef",
32             cpu=1024, # 1 CPU
33             memory_limit_mib=4096, # 4GB RAM
34         )
35
36         # grant permissions
37         table.grant_read_write_data(taskdef.task_role)
38         taskdef.add_to_task_role_policy(
39             iam.PolicyStatement(
40                 effect=iam.Effect.ALLOW,
41                 resources=["*"],
42                 actions=["ssm:GetParameter"]
43             )
44         )
45
46         ⑤
47         container = taskdef.add_container(
48             "EcsClusterQaBot-Container",
49             image=ecs.ContainerImage.from_registry(
50                 "tomomano/qabot:latest"
51             ),
52         )
```

① □□□□

8.4.1. ECS □ Fargate

ECS □ Fargate □□□□□□□□□□□□□□□□□□□□□□□□

```
1 cluster = ecs.Cluster(
2     self, "EcsClusterQaBot-Cluster",
3     vpc=vpc,
4 )
5
6 taskdef = ecs.FargateTaskDefinition(
7     self, "EcsClusterQaBot-TaskDef",
8     cpu=1024, # 1 CPU
9     memory_limit_mib=4096, # 4GB RAM
10 )
11
12 container = taskdef.add_container(
13     "EcsClusterQaBot-Container",
14     image=ecs.ContainerImage.from_registry(
15         "tomomano/qabot:latest"
16     ),
17 )
```

`cluster =` □□□□□□□ ECS □□□□□□□□□□□□□□

□□□ `taskdef=ecs.FargateTaskDefinition` □□□□□ Fargate □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ 1 CPU, 4GB RAM □□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□1□□□□□□□1□□□□□□□□□□□□□□□

□□□□ `container =` □□□□□□□□□□□□□□□□□□ Docker image □□□□□□□□□ □□□□□ Docker Hub □□□□□□□ image □□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□

4096 (4 vCPU)	Between 8 GB and 30 GB in 1-GB increments

8.5. □□□□□□□□□□

□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ SSH □□□□□□□□□□□□□□□□□□□□

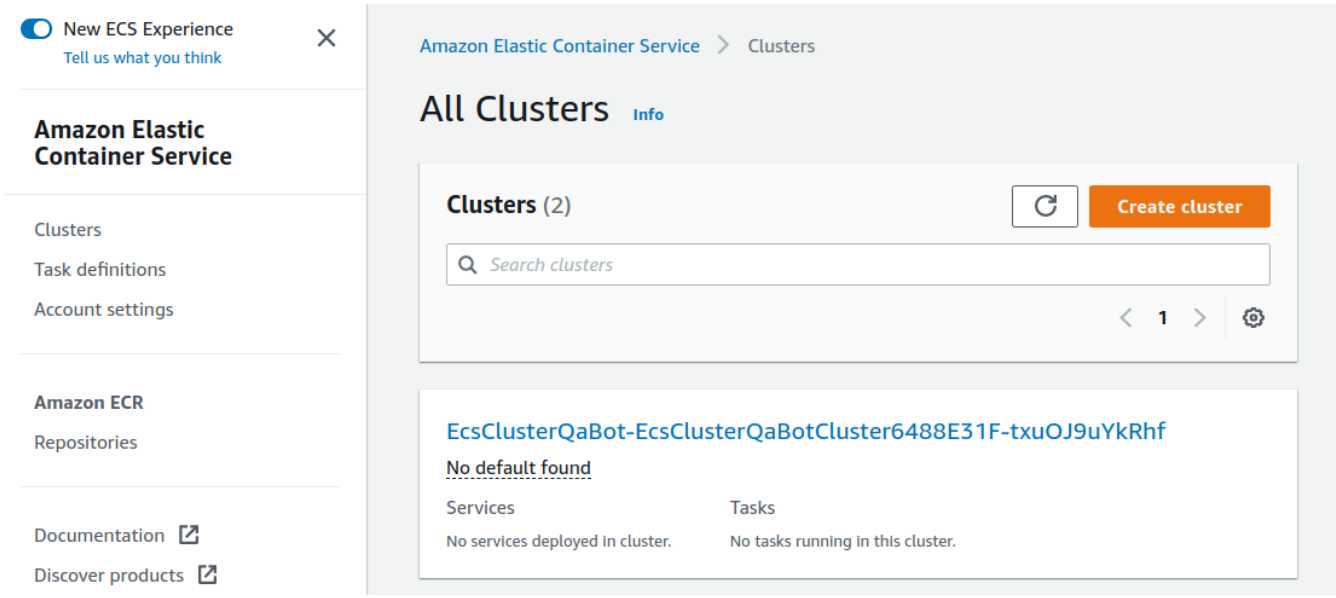


Figure 48. ECS □□□□□□□

□□□□□□□□□□□□□□□□□□□□ Task Definitions □□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□ EcsClusterQaBotEcsClusterQaBotTaskDefXXXX □□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□ Figure 49 □□□□□□□□□□□□□□□□□□□□ □□□□ CPU □□□□□□□□□ Docker container □□□□□□□□□□□□□□□□□ Task Definition □□□□□□□□□□□□□□□□□□

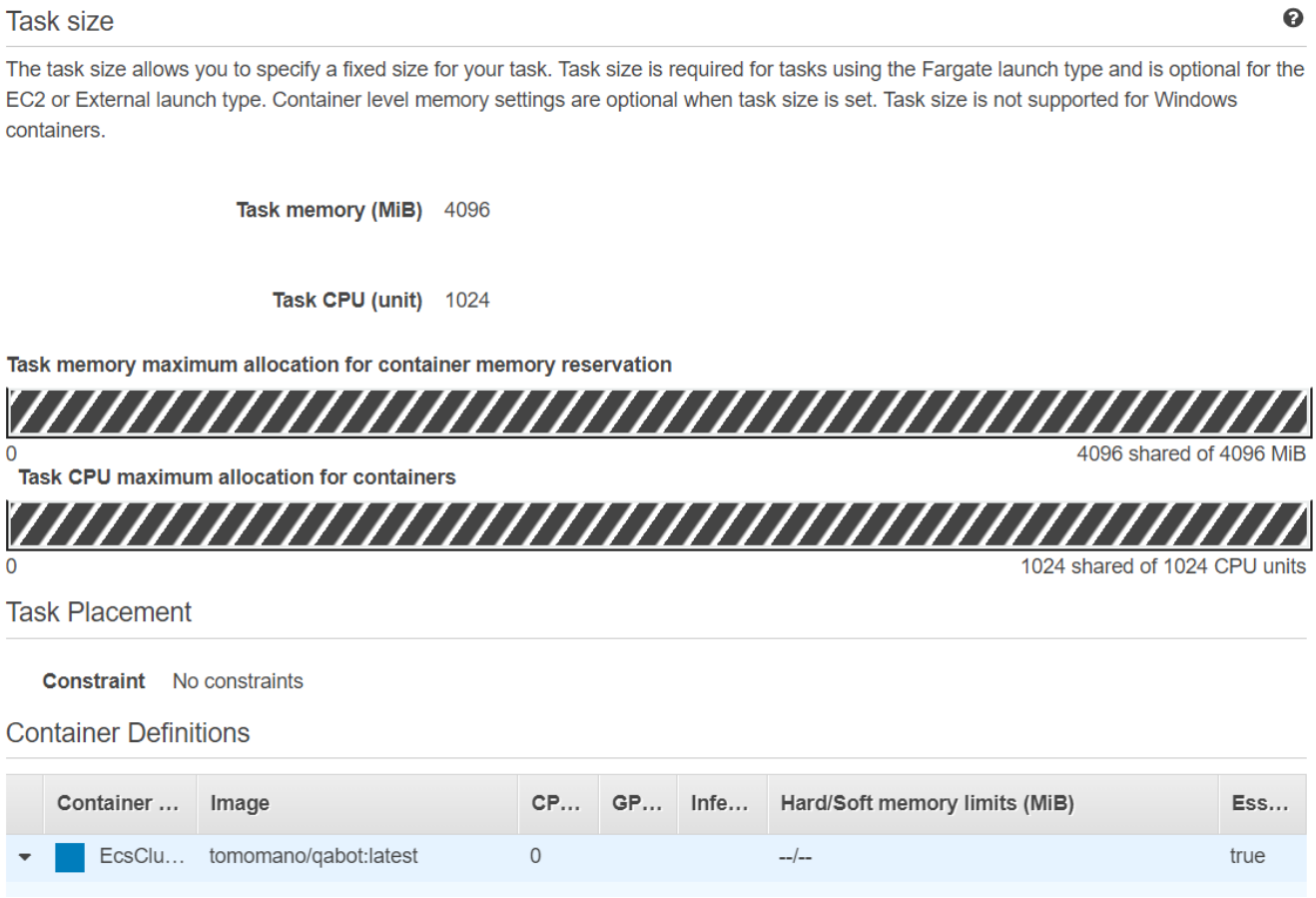


Figure 49. Task definition □□□

8.6. □□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

ECS □□ (`run_task.py`) □□□□□ (handson/qa-bot/run_task.py)□

□□□□□□□□□□□□ECS□□□□□□□□□□□□□□□□□□□□□□□□□□

```
$ python run_task.py ask "A giant peach was flowing in the river. She picked it up and brought it home. Later, a healthy baby was born from the peach. She named the baby Momotaro." "What is the name of the baby?"
```

`run_task.py` □□□□□□□□□ □□□□□□□□□ AWS □□□□□□□□□□□□□□□□□□□□□□□□□□

"ask" □□□□□□□□□□ (context) □□□ (question) □□□□□□□□□□□□□□

□□□□□□□□□□□□□□□ "Waiting for the task to finish..." □□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□ AWS □□ ECS □□□□□□□□□□□□□□□ Fargate □□□□□□□□□□□□□□ Docker □□ AWS □□□□□□□□□□□□□□□□□□□□

48ce33531af149bd8747064ae830535e

Stop

Configuration | Logs | Networking | Tags

Status

Last status	Desired status	Stopped reason	Started at
Deprovisioning	Stopped	Essential container in task exited	2021-06-29T15:30:46.572Z
Created at			
2021-06-29T15:28:20.534Z			

Configuration

Platform version	Task definition	CPU	Memory
1.4.0	EcsClusterQaBotEcsClusterQaBotTaskDef54F4C2A5: 8	1 vCPU	4 GB

Containers (1)

Container name	Image URI	Status	CPU	Memory hard/soft limit
EcsClusterQaBot-Container	tomomano/qabot:latest	Stopped	-	- / -

Figure 51. □□□□□□□□□□□

□□□ `run_task.py` □□□□□□□□□□□□□□□□□□□□□□□□□□□ Figure 52 □□□□□□□□□□□□□□□□□□□□□□□ "Momotaro" □□□□□□□□□□□□□□□□□□□□

```
(.env) tomoyuki@ammonite:qa-bot$ python run_task.py ask "A giant peach was flowing in the river. She pick
ed it up and brought it home. Later, a healthy baby was born from the peach. She named the baby Momotaro.
" "What is the name of the baby?"
Submitting task...
Task ARN: arn:aws:ecs:ap-northeast-1:606887060834:task/EcsClusterQaBot-EcsClusterQaBotCluster6488E31F-txu
OJ9uYkRhf/48ce33531af149bd8747064ae830535e
Waiting for the task to finish...
.............................................................................................
.............................................................
Context: A giant peach was flowing in the river. She picked it up and brought it home. Later, a healthy b
aby was born from the peach. She named the baby Momotaro.
Question: What is the name of the baby?
Answer: Momotaro
Score: 0.9704799652099609
```

Figure 52. □□□□□□□□□□□

8.7. □□□□□□□□□

□□□□□□□□□□□□□□□□□□□

```
$ python run_task.py ask_many
```

□□□□□□□□□□□□□□□□□□□□ ECS □□□□□□□□□□□□□□□□□□□□□□□□ (Figure 53)□ □□□ Fargate □□

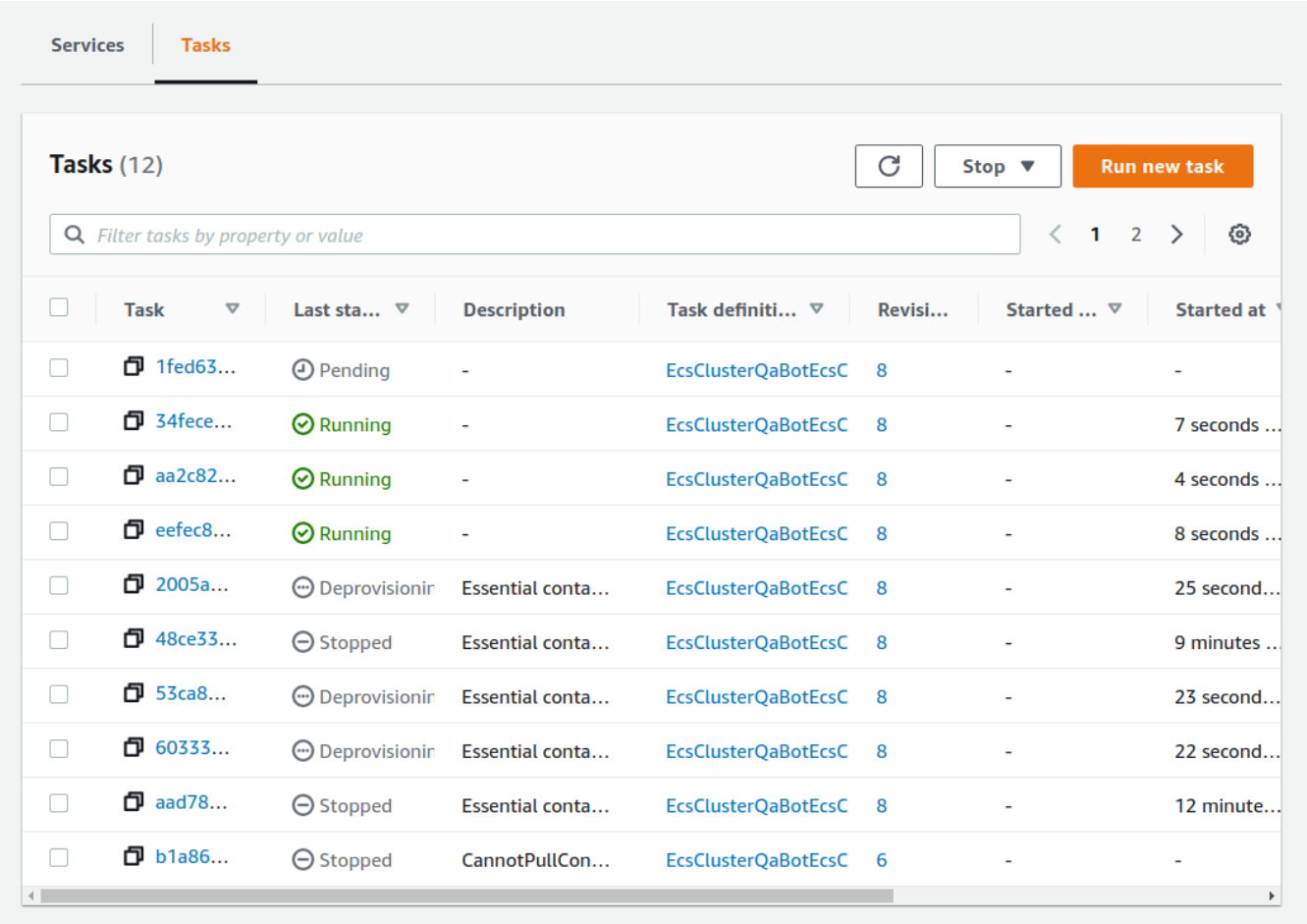


Figure 53. □□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□ "STOPPED" □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□

```
$ python run_task.py list_answers
```

□□□□□□□ Figure 54 □□□□□□□□□□□□□□□□□□ □□□

```
Context: Nikola Tesla (Serbian Cyrillic: ??????????? ??????????; 10 July 1856 ??? 7 January 1943) was a Serbian American inven
tor, electrical engineer, mechanical engineer, physicist, and futurist best known for his contributions to the design of the mo
dern alternating current (AC) electricity supply system.
Question: In what year did Tesla die ?
Answer: 1943)
Score: 0.47624243081909157
2
Context: The Normans (Norman: Nourmands; French: Normands; Latin: Normanni) were the people who in the 10th and 11th centuries
gave their name to Normandy, a region in France. They were descended from Norse ("Norman" comes from "Norseman") raiders and pi
rates from Denmark, Iceland and Norway who, under their leader Rollo, agreed to swear fealty to King Charles III of West Franci
a. Through generations of assimilation and mixing with the native Frankish and Roman-Gaulish populations, their descendants wou
ld gradually merge with the Carolingian-based cultures of West Francia. The distinct cultural and ethnic identity of the Norman
s emerged initially in the first half of the 10th century, and it continued to evolve over the succeeding centuries.
Question: What century did the Normans first gain their separate identity ?
Answer: 10th
Score: 0.7034500812467961
3
Context: The Normans (Norman: Nourmands; French: Normands; Latin: Normanni) were the people who in the 10th and 11th centuries
gave their name to Normandy, a region in France. They were descended from Norse ("Norman" comes from "Norseman") raiders and pi
rates from Denmark, Iceland and Norway who, under their leader Rollo, agreed to swear fealty to King Charles III of West Franci
a. Through generations of assimilation and mixing with the native Frankish and Roman-Gaulish populations, their descendants wou
ld gradually merge with the Carolingian-based cultures of West Francia. The distinct cultural and ethnic identity of the Norman
s emerged initially in the first half of the 10th century, and it continued to evolve over the succeeding centuries.
Question: Who was the Norse leader ?
Answer: Rollo,
Score: 0.9961329869148798
```

Figure 54. `$ python run_task.py list_answers` □□□□□

□□□□□□ □□□□□□□

Chapter 9. Hands-on #4: AWS Batch □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□ ECS □ Fargate □□□□□□□□□□□□□□□□□□□□

Figure 55. AWS Batch □□□□□

□□□□□□□□□□□ ECS □ ASG □□□□□□□□□□□□

	EC2	Fargate
GPU	Yes	No
Launch speed	Slow	Fast
Task placement flexibility	Low	High
Programming complexity	High	Low

□□□□□□□□□□□□□□ EC2 □□□□ CPU □□□□□□□□□□□□□□□□□□□ GPU □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□ Fargate □□□□□□□□□□□□□□ CPU □□4□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□

```
$ docker run mymnist --lr 0.1 --momentum 0.5 --epochs 10
```

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ (--lr □□□□□□□□□□□ --momentum □□□□□□□□□□□□□) □□□□□□□□□□□□□□□□□□□□

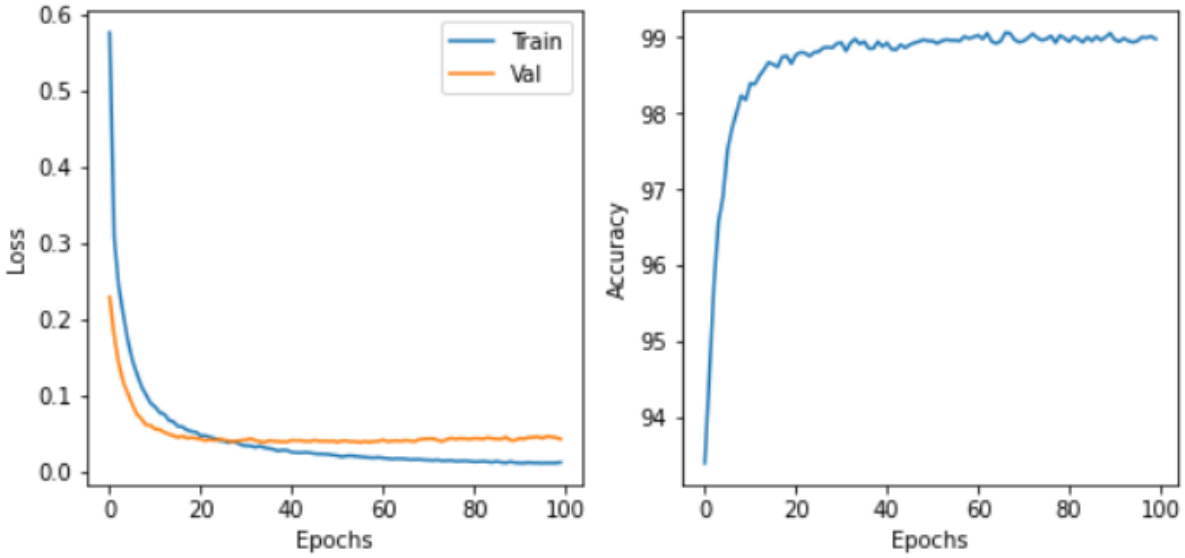


Figure 58. (□) Train/Validation □□□□□□□□ Loss □□□□□□□□□□□ (□) Validation □□□□ Accuracy □□□□□□□□□□

□□□□□□□□□□

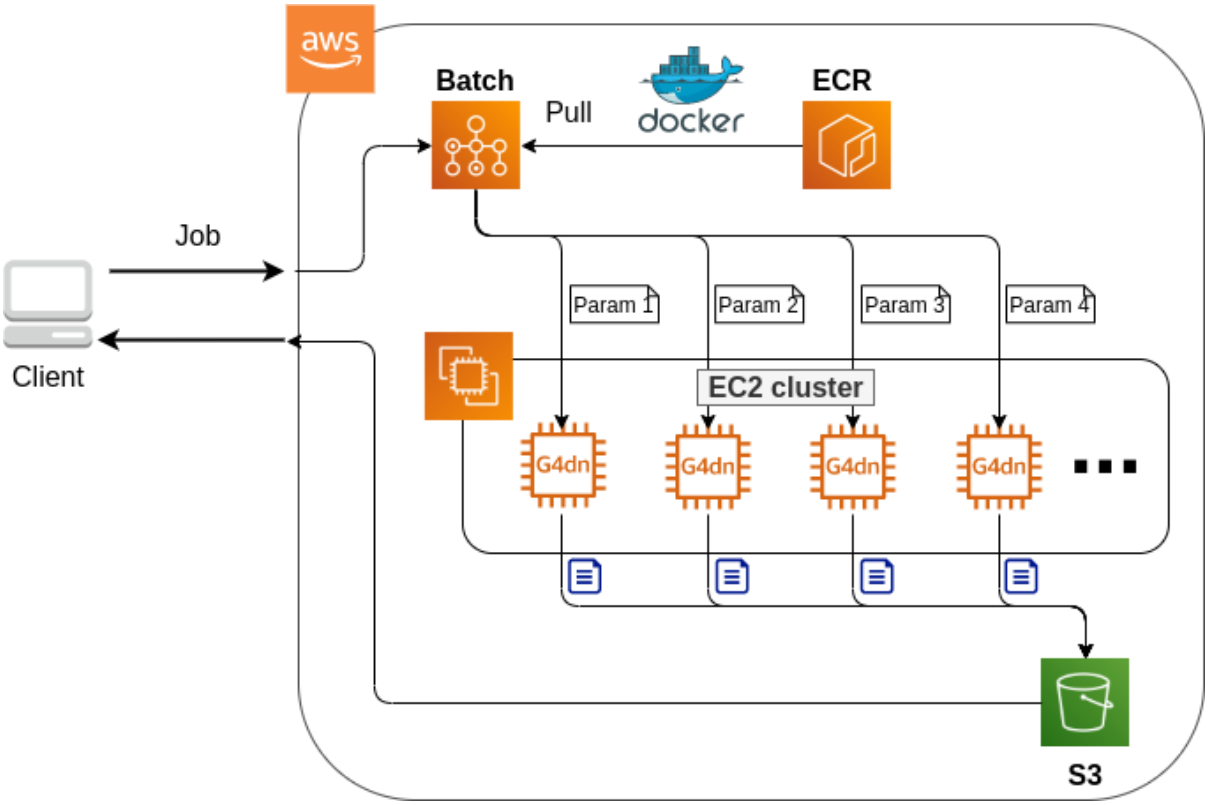


Figure 59. □□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□

- □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ Batch □□□□□□□□□□
- Batch □□□□□□□□□□□□ EC2 □□□□□□□□□□□□□□□□□□□□
- □□□□□□□□□ `g4dn.xlarge` □□□□□□□□□□□□
- Docker □□□□□□ AWS □□□□□□□□ ECR (Elastic Container Registry) □□□□□□□□
- □□
- □□□□□□□□□□□□□□□ S3 □□□□□□□
- □□□□□□□□□□□ S3 □□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ (handson/aws-batch/app.py)□

```
class SimpleBatch(Stack):

    def __init__(self, scope: Construct, construct_id: str, **kwargs) -> None:
        super().__init__(scope, construct_id, **kwargs)

        ①
        bucket = s3.Bucket(
            self, "bucket",
            removal_policy=cdk.RemovalPolicy.DESTROY,
            auto_delete_objects=True,
        )

        vpc = ec2.Vpc(
```

```
14            self, "vpc",
15            # other parameters...
16        )
17
18        ②
19        managed_env = batch_alpha.ComputeEnvironment(
20            self, "managed-env",
21            compute_resources=batch_alpha.ComputeResources(
22                vpc=vpc,
23                allocation_strategy=batch_alpha.AllocationStrategy.BEST_FIT,
24                desiredv_cpus=0,
25                maxv_cpus=64,
26                minv_cpus=0,
27                instance_types=[
28                    ec2.InstanceType("g4dn.xlarge")
29                ],
30            ),
31            managed=True,
32            compute_environment_name=self.stack_name + "compute-env"
33        )
34
35        ③
36        job_queue = batch_alpha.JobQueue(
37            self, "job-queue",
38            compute_environments=[
39                batch_alpha.JobQueueComputeEnvironment(
40                    compute_environment=managed_env,
41                    order=100
42                )
43            ],
44            job_queue_name=self.stack_name + "job-queue"
45        )
46
47        ④
48        job_role = iam.Role(
49            self, "job-role",
50            assumed_by=iam.CompositePrincipal(
51                iam.ServicePrincipal("ecs-tasks.amazonaws.com")
52            )
53        )
54        # allow read and write access to S3 bucket
55        bucket.grant_read_write(job_role)
56
57        ⑤
58        repo = ecr.Repository(
59            self, "repository",
60            removal_policy=cdk.RemovalPolicy.DESTROY,
61        )
62
63        ⑥
64        job_def = batch_alpha.JobDefinition(
```

```
            self, "job-definition",
            container=batch_alpha.JobDefinitionContainer(
                image=ecs.ContainerImage.from_ecr_repository(repo),
                command=["python3", "main.py"],
                vcpus=4,
                gpu_count=1,
                memory_limit_mib=12000,
                job_role=job_role,
                environment={
                    "BUCKET_NAME": bucket.bucket_name
                }
            ),
            job_definition_name=self.stack_name + "job-definition",
            timeout=cdk.Duration.hours(2),
        )
```

① □□□□□□□□□□□□□□□□ S3 □□□□□□□□□□□□□

② □□ Compute environment □□□□□□□□□ □□□□ `g4dn.xlarge` □□□□□□□□□□□□□□□□□□□□□□□□□□ vCPU □□□□ 64 □□□□□□□□□□□□□□□□ vCPU □ 0 □□□□ □□□□□□□□□□□□□□□□□□□□

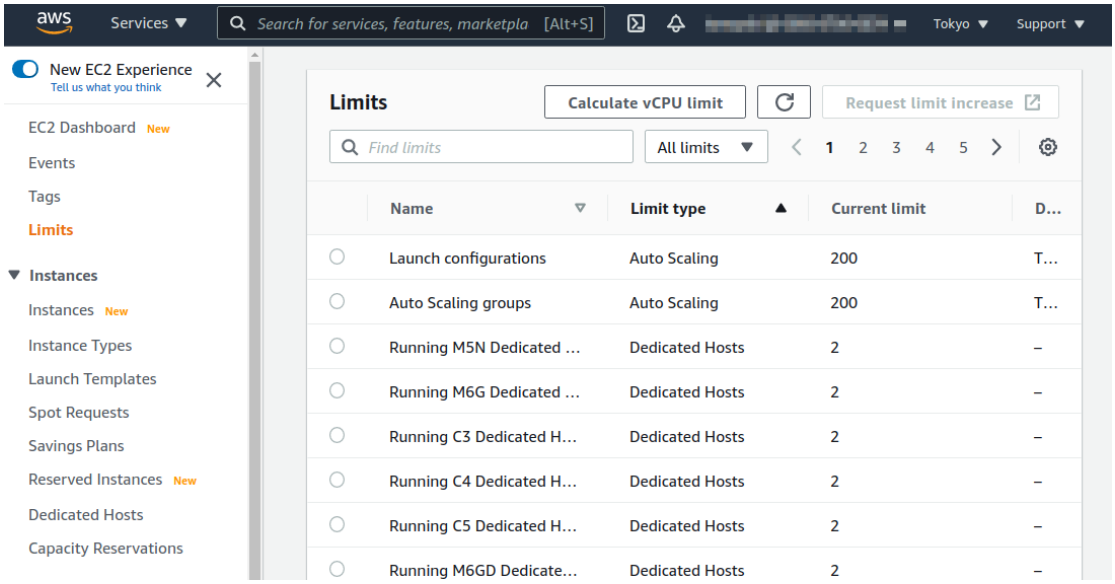


Figure 60. EC2□□□□□□□□□□□□□□□□□□□□

9.6. □□□□□□□□□□

□□□□□□□□□□□□□□□□□□□

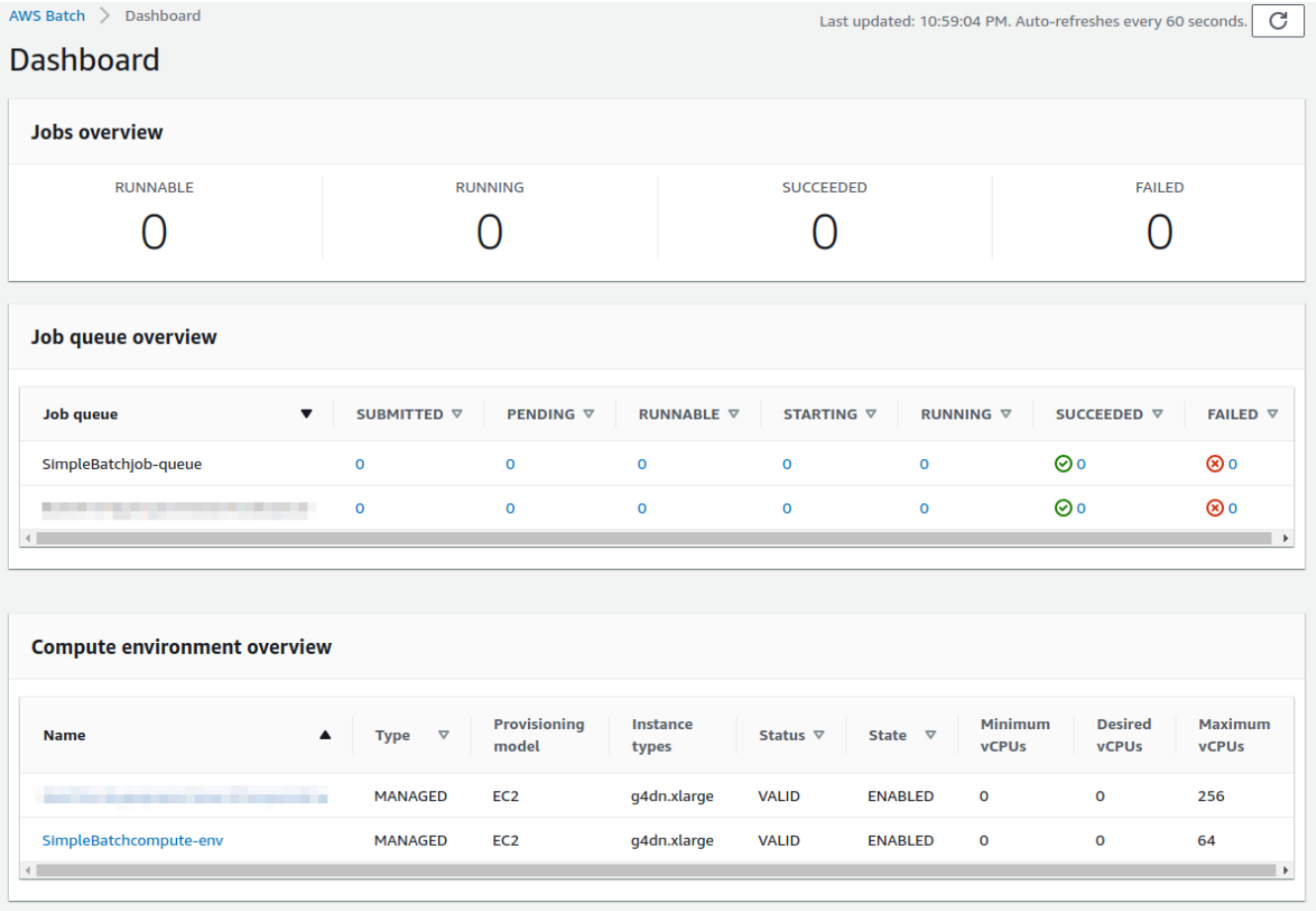


Figure 61. AWS Batch □□□□□□□□ (□□□□□□□)

□□□□□□□□□□□□□□□□□□□□□□□□□□□□ Compute environment overview □□□ SimpleBatchcompute-env □□□□□□□□□□□ Compute environment □□□□□□□□□□□□□□□□□□□□□□□□□□□□ (□□□□□□□□□□□□□□□□□) □□□□ □□□□□□□□□□□□□□□□ g4dn.xlarge □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□ Minimum vCPUs □0□Maximum vCPUs □ 64 □□□□□□□□□□□□□□□□□□□□

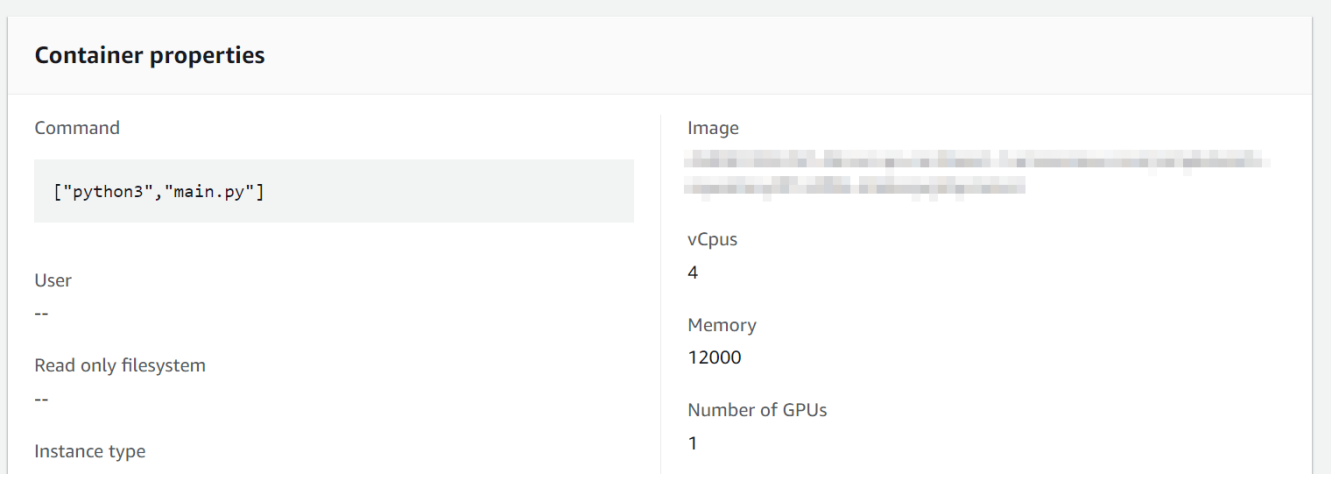


Figure 62. AWS Batch □□ Job definition □□□

9.7. Docker image □ ECR □□□□□

□□□ Batch □□□□□□□□□□□□□□□□□□□□□□□□□□□□ Docker □□□□□□□□□□□□ (pull) □□□□□□□□□□□ □□□□□□□□□ (Chapter 8) □□□□□□□□□□□□□□ Docker Hub □□□□□□□□ pull □□□□□ □□□□□□□□□□□□□ AWS □□□□□□□□□□□□□□□□□□□□ **ECR (Elastic Container Registry)** □ image □□□□□□□□□□□□□□□□□□ ECR □□□□□□□□□□□□□□□□□□□□

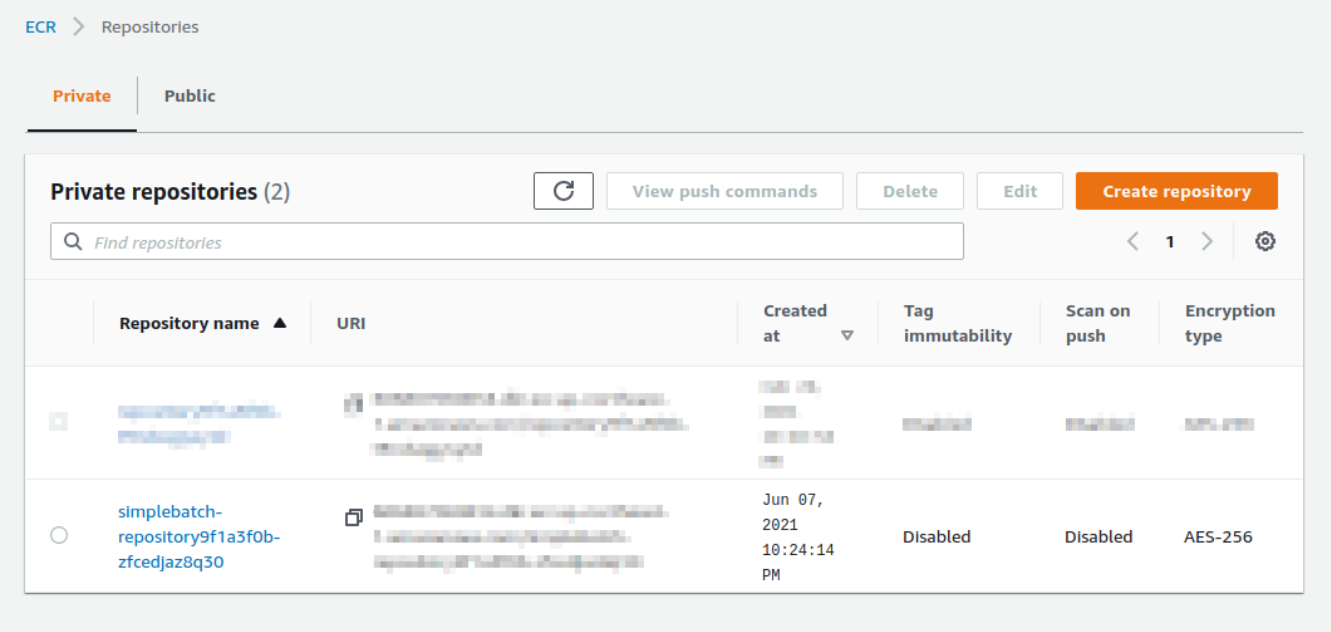


Figure 63. ECR □□□□□□□□

□□□□□□□

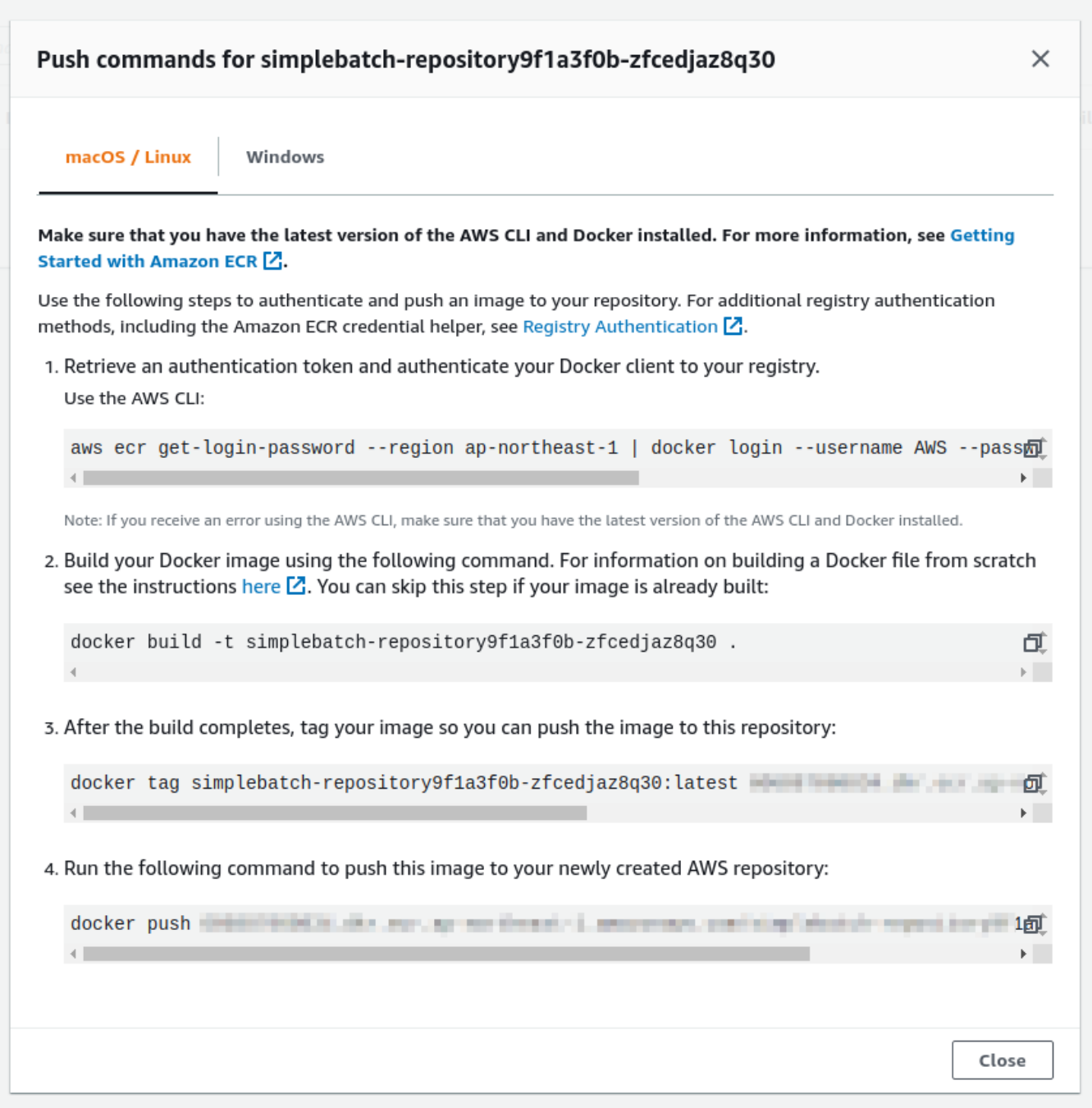


Figure 64. ECR □□ push □□□□

□□□

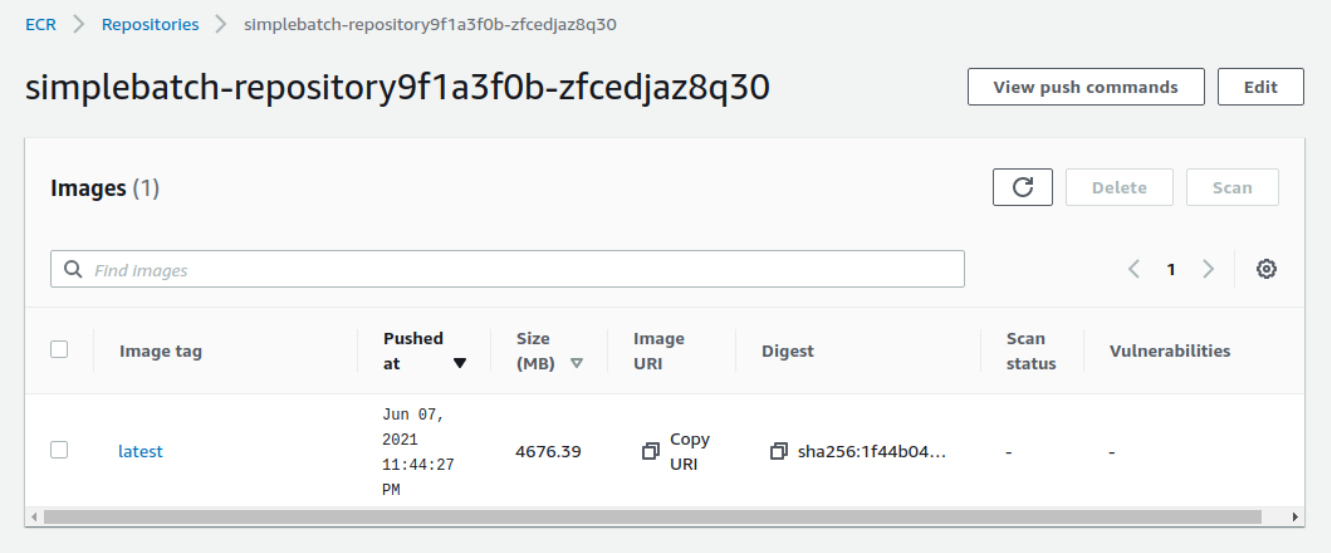


Figure 65. ECR □ image □□□□□□□□□

9.8. □□□□□□□□□□□□

□□□□□□□□□□□□ AWS Batch □□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□ `notebook/` □□□□□□□□□□□□□□□ run_single.ipynb □□□□□□□□□□□□□□□□□□□□ (`.ipynb` □ Jupyter notebook □□□□□□□□)□ □□□ Jupyter notebook □□□□□□□

□□□□□□□□□□□□ `venv` □□□□□□□□□□□ Jupyter notebook □□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□ Jupyter notebook □□□□□□□□

```
# .env □□□□□□□□□□□□□□□
(.env) $ cd notebook
(.env) $ jupyter notebook
```

Jupyter notebook □□□□□□□□ `run_single.ipynb` □□□□

□□□ [1], [2], [3] □□□□□□□□□□□□□□□□□□□□□□□□□□□ (`submit_job()`) □□□□□□□□□

```
 1 # [1]
 2 import boto3
 3 import argparse
 4
 5 # [2]
 6 # AWS □□□□□□ ...□□...
 7
 8 # [3]
 9 def submit_job(lr:float, momentum:float, epochs:int, profile_name="default"):
10     if profile_name is None:
11         session = boto3.Session()
12     else:
13         session = boto3.Session(profile_name=profile_name)
14     client = session.client("batch")
```

```

    title = "lr" + str(lr).replace(".", "") + "_m" + str(momentum).replace(".", "")
    resp = client.submit_job(
        jobName=title,
        jobQueue="SimpleBatchjob-queue",
        jobDefinition="SimpleBatchjob-definition",
        containerOverrides={
            "command": ["--lr", str(lr),
                        "--momentum", str(momentum),
                        "--epochs", str(epochs),
                        "--uploadS3", "true"]
        }
    )
    print("Job submitted!")
    print("job name", resp["jobName"], "job ID", resp["jobId"])
```

submit_job() □□□□□□□□□□□□□□□□ Section 9.4 □□ MNIST □ Docker □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

```
$ docker run -it mymnist --lr 0.1 --momentum 0.5 --epochs 10
```

□□□□ --lr 0.1 --momentum 0.5 --epochs 10 □□□□□□□□□□□□□□□□□□□□□□□□□□

AWS Batch □□□□□□□□□□□□□ContainerOverrides □ command □□□□□□□□□□□□□□□□□□□□

[4] □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ AWS □□□□□□□□□□□□□□□□ AWS Batch □□□□□□□□□□□□□□□ Figure 66 □□□□□□□□□□□□□□□□□□

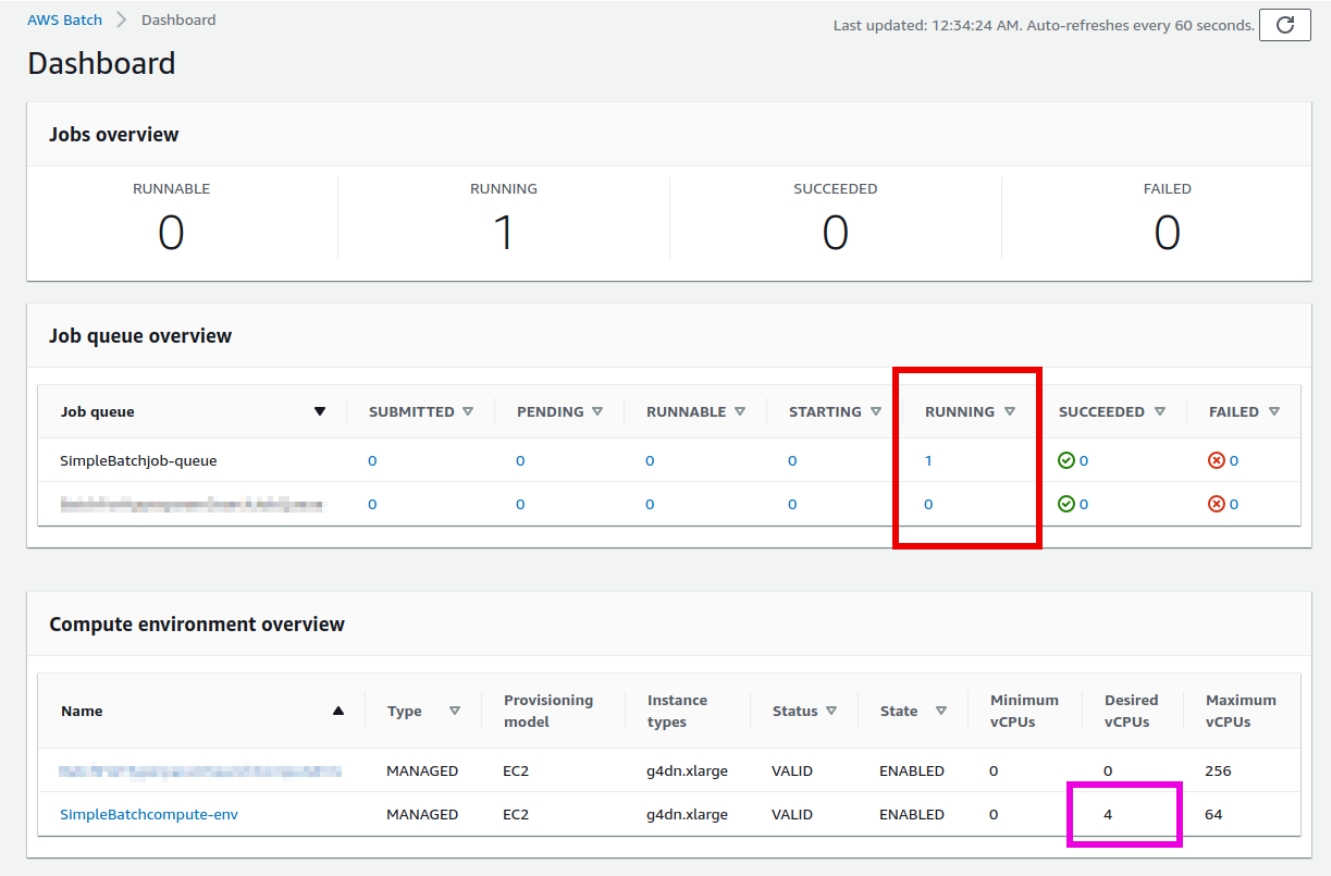


Figure 66. AWS Batch □□□□□□□□□□□□□□□□

Figure 66 □□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□ SUBMITTED □□□□□□□□□ RUNNABLE □□□□□□□□□□□□ RUNNABLE □□□□ □□□□□□□□□□□□□□□□□□□□□ Compute environment □□□□□□□□□□□□□□□□□□□□

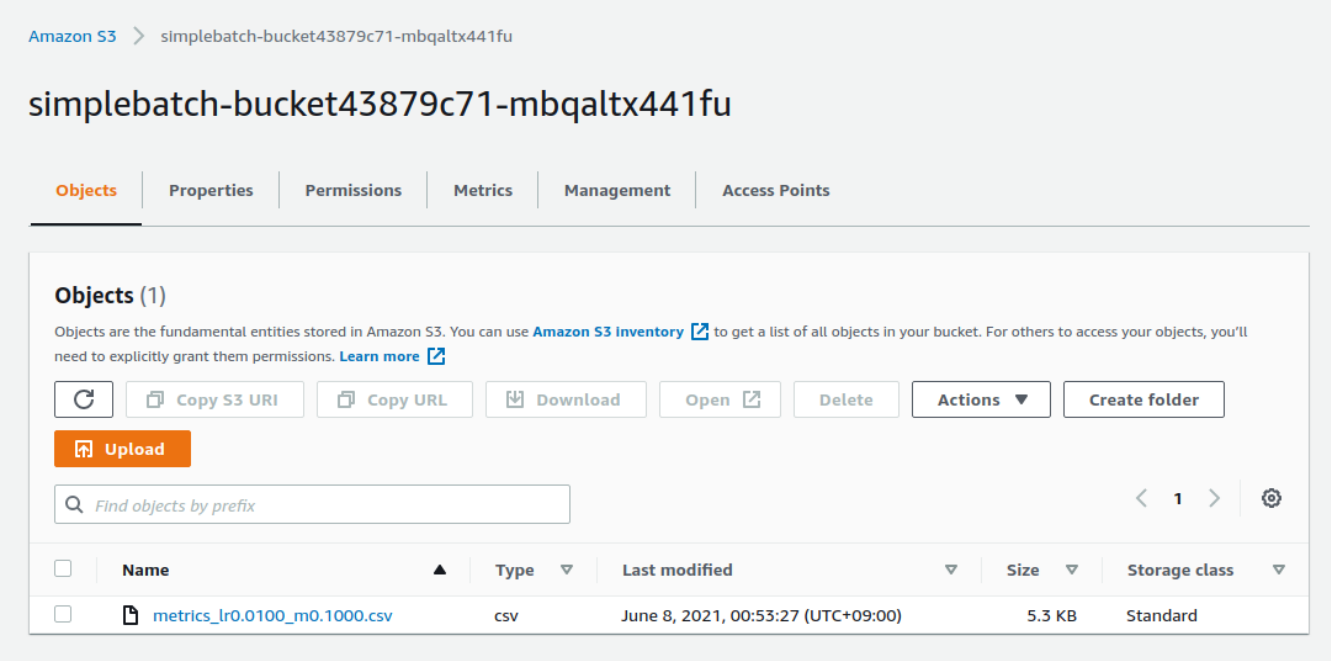


Figure 67. □□□□□□□□□□ *S3* □□□□□□

□□□□□□ `run_single.ipynb` □□□□□□□□□ [5] □□ [7] □□□□□□□□□□□□□ CSV □□□□□□□□□□□□□□□□□□□□

```
# [5]
import pandas as pd
import io
from matplotlib import pyplot as plt

# [6]
def read_table_from_s3(bucket_name, key, profile_name=None):
    if profile_name is None:
        session = boto3.Session()
    else:
        session = boto3.Session(profile_name=profile_name)
    s3 = session.resource("s3")
    bucket = s3.Bucket(bucket_name)

    obj = bucket.Object(key).get().get("Body")
    df = pd.read_csv(obj)

    return df

# [7]
bucket_name = "simplebatch-bucket43879c71-mbqaltx441fu"
df = read_table_from_s3(
    bucket_name,
    "metrics_lr0.0100_m0.1000.csv"
)
```

[6] □ S3 □□ CSV □□□□□□□□□□□□ pandas □ `DataFrame` □□□□□□□□□□□□□□□□□□□□□□□□□□ [7] □□□□□□□ `bucket_name` □□□□□□□□□□□□□□□□□□□□

□□□□ [9] □□□□□□ CSV □□□□□□□□□□□□□□ (Figure 68)□ □□□□□□□□□□□□□□□□□□□ AWS Batch □□□□ MNIST □□□□□□□□□□□□□□□□□□

```
# [9]
fig, (ax1, ax2) = plt.subplots(1,2, figsize=(9,4))
x = [i for i in range(df.shape[0])]
ax1.plot(x, df["train_loss"], label="Train")
ax1.plot(x, df["val_loss"], label="Val")
ax2.plot(x, df["val_accuracy"])

ax1.set_xlabel("Epochs")
ax1.set_ylabel("Loss")
ax1.legend()

ax2.set_xlabel("Epochs")
ax2.set_ylabel("Accuracy")

print("Best loss:", df["val_loss"].min())
print("Best loss epoch:", df["val_loss"].argmin())
print("Best accuracy:", df["val_accuracy"].max())
print("Best accuracy epoch:", df["val_accuracy"].argmax())
```

```
Best loss: 0.2320499013662338
Best loss epoch: 0
Best accuracy: 99.15833333333332
Best accuracy epoch: 67
```

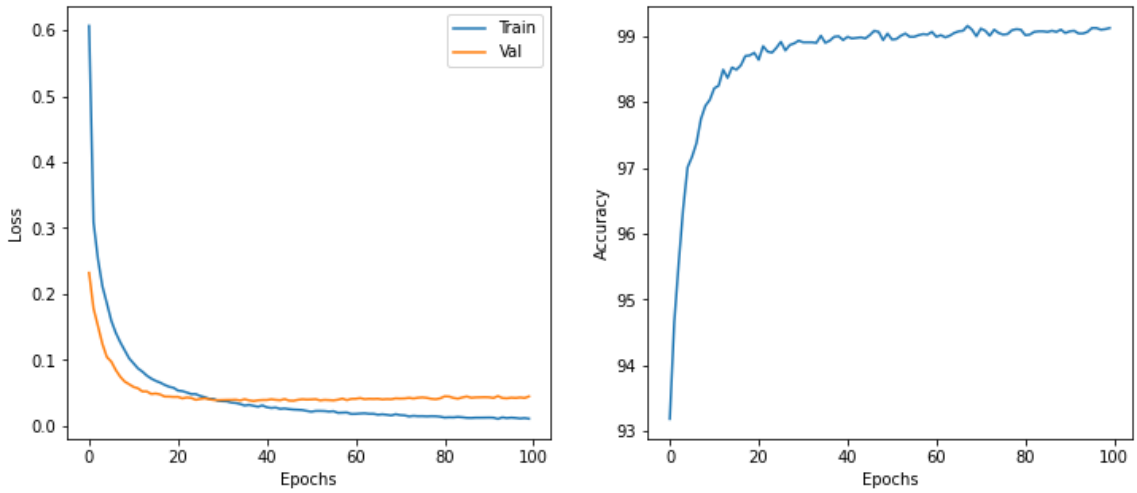


Figure 68. AWS Batch □□□□ MNIST □□□□□□□□□□

9.9. □□□□□□□ Job □□□□□□

□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□ AWS Batch □□

□□□□□□□□ `run_single.ipynb` □□□□□□□□□□□□□□ `run_sweep.ipynb` □□□□

□□ [1], [2], [3] □ `run_single.ipynb` □□□□□□□□

```
# [1]
```

```
import boto3
import argparse

# [2]
# AWS □□□□□□ ...□□...

# [3]
def submit_job(lr:float, momentum:float, epochs:int, profile_name=None):
    # ...□□...
```

□□ [4] □ for □□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ 9□□□□□□□□□□ RUNNING □□□□□□□□□□□□□□□□□□□□

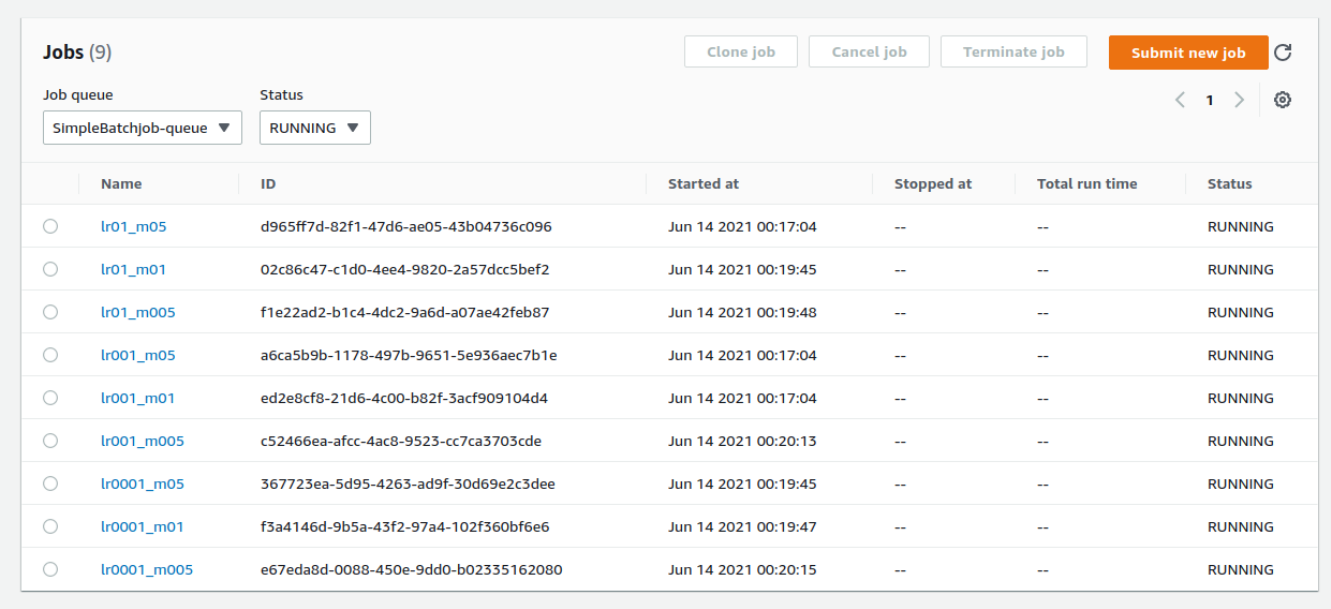


Figure 70. □□□□□□□□□□□□□□□□□ Job □□

□□□ EC2 □□□□□□□□□□□□ □□□□□□□□ Instances □□□□□□□ Figure 71 □□□□□□□□□□□□□□□□□□□□□□□□□□□□□ g4dn.xlarge □ 9 □□□□□□□□□□□□□□□□ Batch □□

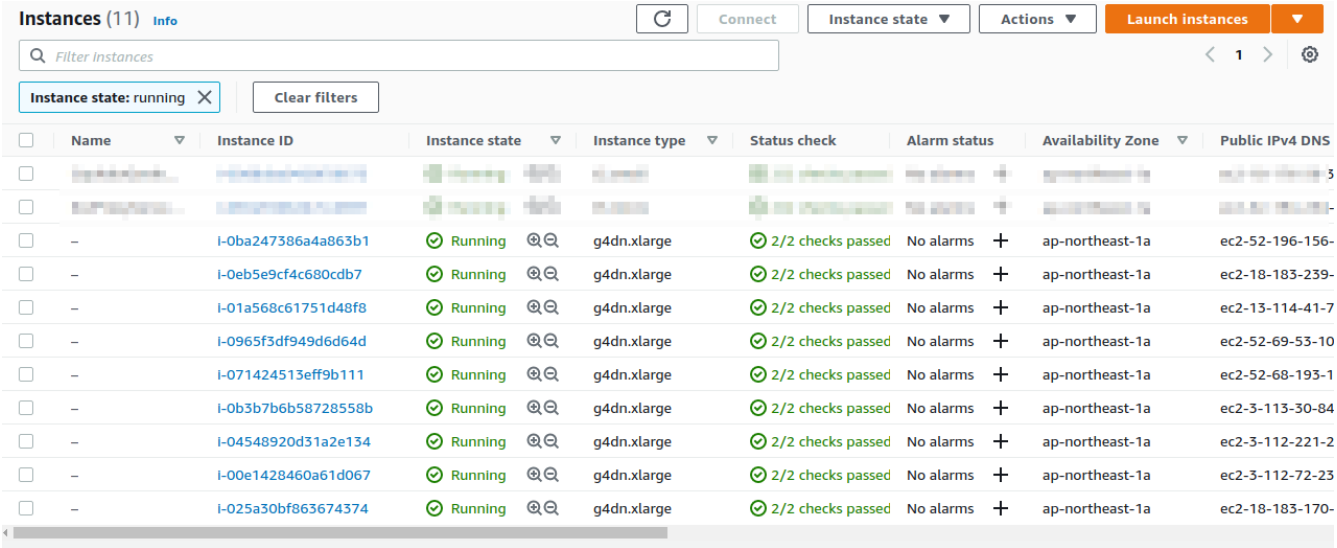


Figure 71. □□□□□□□□□□□□□□□□□ EC2 □□□□□□□□□□

□□□□□□□□□□□□□□□□□ Job □□□□□□□□□□□□□□□ (□□□□ 10-15 □□□□□□□□□)□ □□□□□□□□□□□□□□□□□□□□□□□□□ SUCCEEDED □ 9 □□□□□□□□□□□ □□□ Compute environment □ Desired vCPUs □ 0 □□□□□□□□□□□□□□□□□ □□□ EC2 □□□□□□□□□□□□□□□□ g4dn □□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□ AWS Batch □□□□□□□□□□□□□□□□□□□□□□□ **EC2** □□□□□□□□□□□□□□□□□□□□

```python
import pandas as pd
import numpy as np
import io
from matplotlib import pyplot as plt

# [6]
def read_table_from_s3(bucket_name, key, profile_name=None):
    if profile_name is None:
        session = boto3.Session()
    else:
        session = boto3.Session(profile_name=profile_name)
    s3 = session.resource("s3")
    bucket = s3.Bucket(bucket_name)

    obj = bucket.Object(key).get().get("Body")
    df = pd.read_csv(obj)

    return df

# [7]
grid = np.zeros((3,3))
for (i, lr) in enumerate([0.1, 0.01, 0.001]):
    for (j, m) in enumerate([0.5, 0.1, 0.05]):
        key = f"metrics_lr{lr:0.4f}_m{m:0.4f}.csv"
        df = read_table_from_s3("simplebatch-bucket43879c71-mbqaltx441fu", key)
        grid[i,j] = df["val_accuracy"].max()

# [8]
fig, ax = plt.subplots(figsize=(6,6))
ax.set_aspect('equal')

c = ax.pcolor(grid, edgecolors='w', linewidths=2)

for i in range(3):
    for j in range(3):
        text = ax.text(j+0.5, i+0.5, f"{grid[i, j]:0.1f}",
                       ha="center", va="center", color="w")
```

□□□□□□□□□□□□□□□□ Figure 72 □□□□

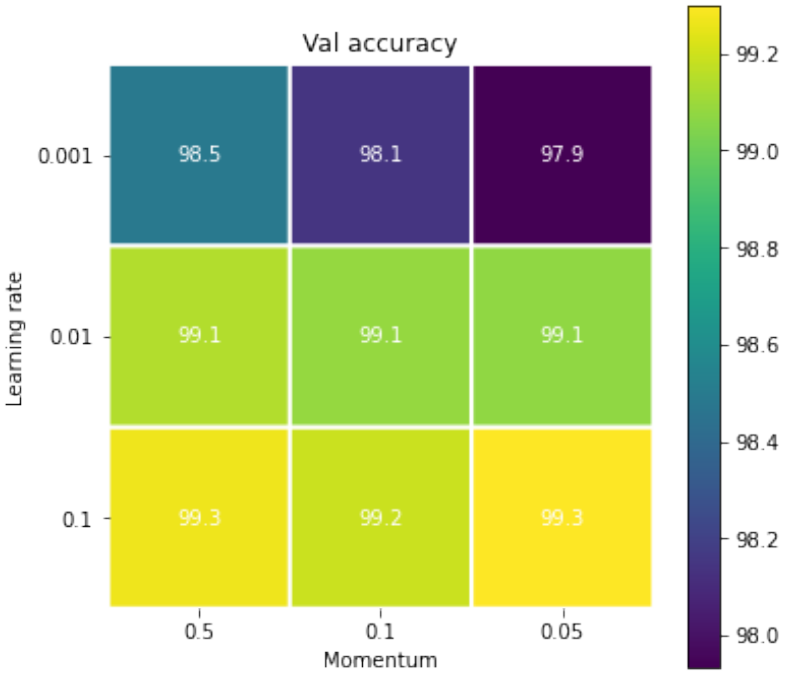


Figure 72. □□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□ 0.1 □□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□ 0.1 □□□□□□□□□□□□□□□□□□□□

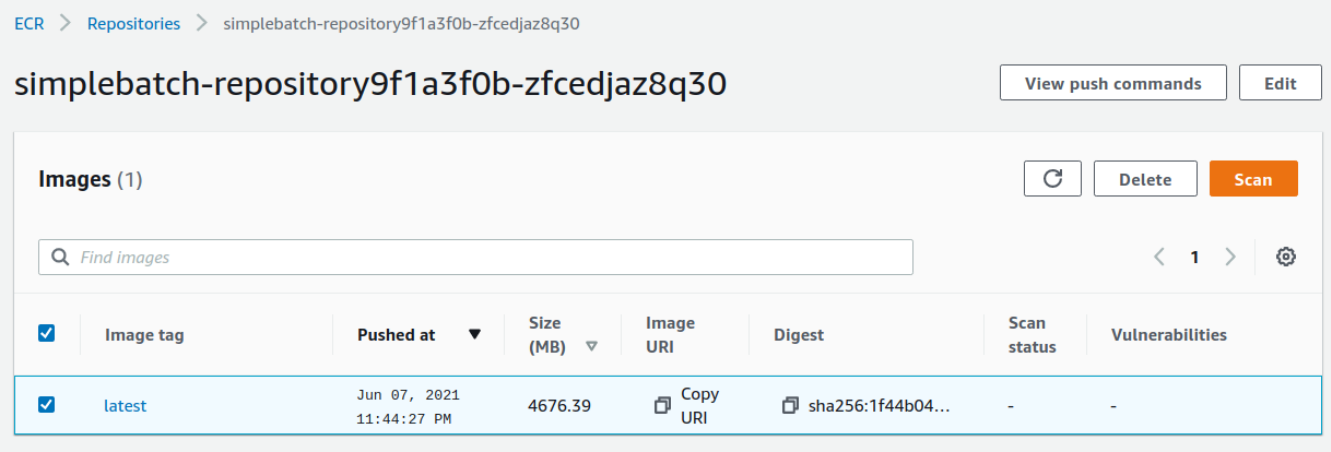


Figure 73. ECR □□ Docker image □□□□□

□□□□□ AWS CLI □□□□□□□□□□□□□□□□□□□□□□□□□□□□ (XXXX □□□□ ECR □□□□□□□□□□□□□□)□

```
$ aws ecr batch-delete-image --repository-name XXXX --image-ids imageTag=latest
```

image □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

```
$ cdk destroy
```

[sec:batch_development_and_debug] === □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□ AWS Batch □□□□□□□□□□□□□□□□□□□□

□□□

Chapter 10. Web □□□□□□□□□

□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□

10.2. REST API

API (Application Programming Interface) □□□□

> Facebook □□□□□□□□□□□□□□□□ GraghQL Foundation □□□□□□□□□□□□□□□□ GraphQL □□□□□□□□□□□□□□□□ REST □□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□ API □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□

Chapter 11. Serverless architecture

□□□□□□□□□□□□□□ (Serverless architecture) □□□□ □□□□□□□□□□□□□□□□ (Serverless computing) □□□□□□□□□□□□□□□□

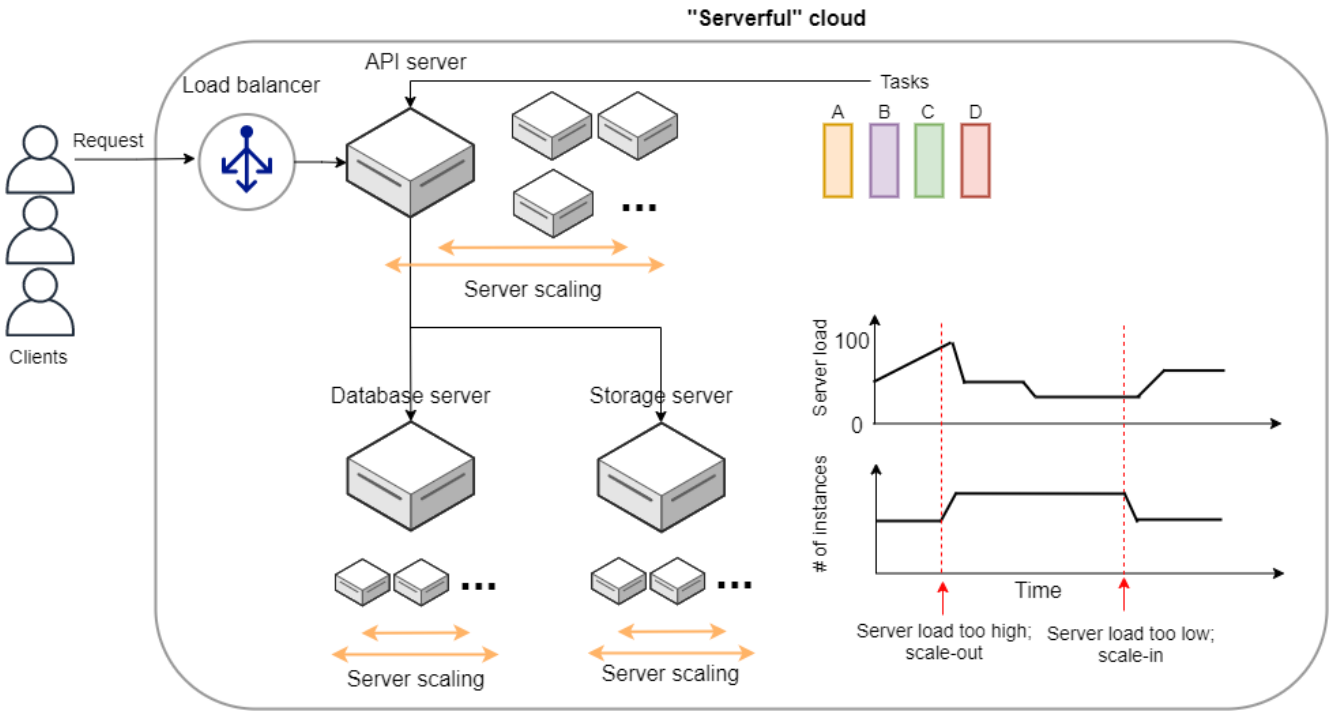


Figure 77. Serverful □□□□□□□□□□

11.2. Serverless □□□□□

Section 11.1 □□□□□□□□□□

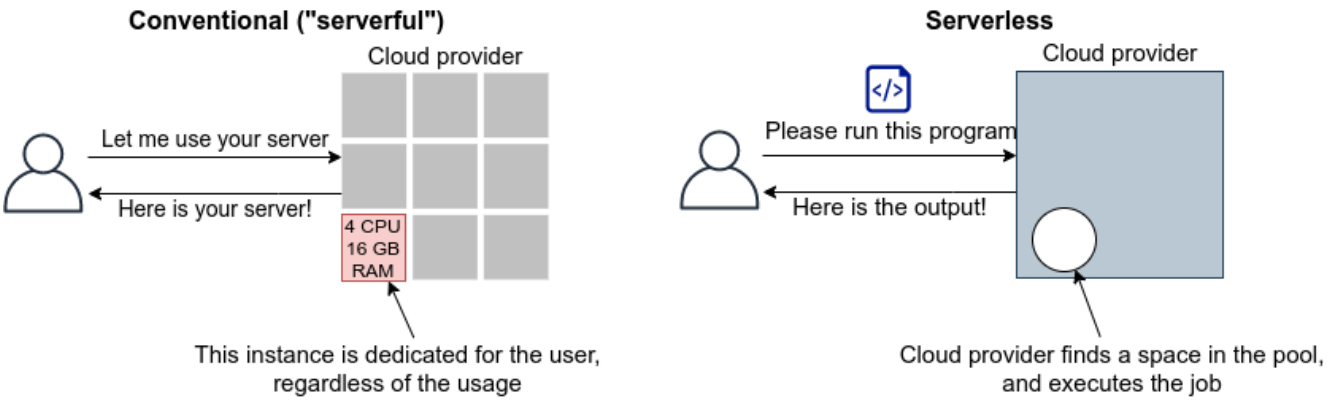


Figure 78. □□□□□□□□ Serverless □□□□□□□

□□□□□□□

Lambda □□□□□□□□□□□□□□□□□□□□□□□□ (Function) □□□□ □□□□□□□□□□□□□□□□□invoke □□□□□ Lambda □□□□□□□ Lambda □□□ invoke □□□□□□□□□□□□□□□□□ (□□□□□□□□□□□□□□□□□□□□□□□□□□□□□) □□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

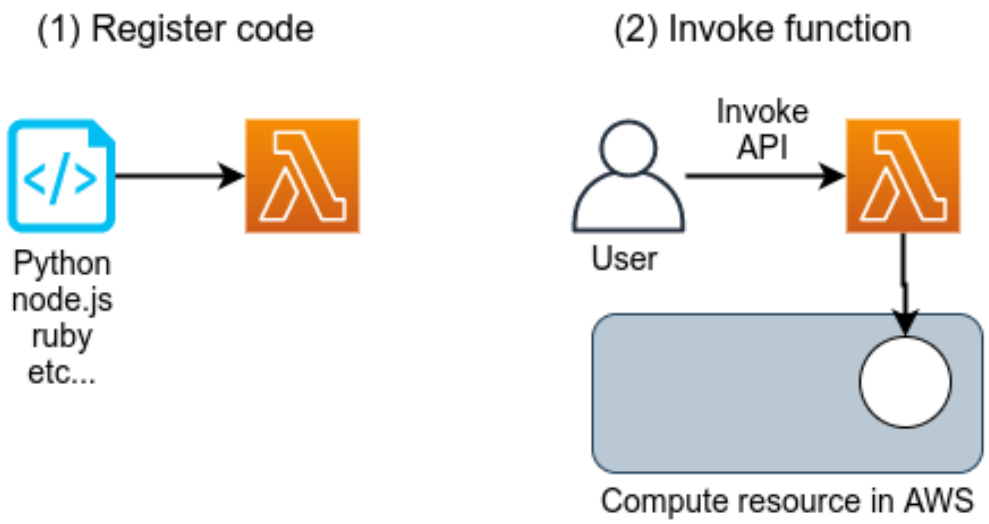


Figure 80. AWS Lambda

□□□□□□□Lambda □□□□□□□□□□□□□□□□□□□□

□□

11.3.2. □□□□□□□□□□□□: S3

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□ (□□□□□□□□□) □□□□□□□□□□□□□□□□ OS □□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ CPU □□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□

□□	□□
Data Transfer OUT From Amazon S3 To Internet	$0.09 per GB

□□□□□□□□□□□□□ $0.025 per GB □□□□□□□□□□□□□□□ □□□□□□1000GB □□□□□ S3 □□□□□□□□□□□□ $25 □□□□□□□□□□□□□□□□ □□□PUT, COPY, POST □□□□□□□□ (=□□□□□□□□□□□) □□□□□□□□□□□□□□□□□□1000□□□□ $0.005 □□□□□□□□□□ GET, SELECT □□□□□□□□ (=□□□□□□□□□□□) □□□□□□1000□□□□ $0.0004 □□□□□□□□□□□ □□□ S3 □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□S3 □□□□□□□□□□□□□□□□□□□□□□□□□ (data-out) □□□ $0.09 per GB □□□□□□□□□□□ □□□□□□□□□□□□□□□□ S3 □□□□ (data-in) □□□□□□□□□□□ □□□ AWS □ □□ Region □□□□□□ (Lambda □ EC2 □□) □□□□□□□□□□□□□□□□□□ AWS □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□

- API Gateway: API □□□□□□□□□□□□□□□□□□ Chapter 13 □□□□□□□
- Fargate: Chapter 8 □□□□ Fargate □□□□□□□□□□□□□□□□□□□□□□□□□ Lambda □□□□□□Lambda □□□□□□□□□□□□□ CPU □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
- Simple Notification Service (SNS): □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
- Step Functions: □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

> □□□□□□□□□□□□□□□□□□□
>
> □□□□□□□□□□□□□□ NO □□□□□□□□□
>
> □□□□□□□□□□□□□□□□□□□□

Chapter 12. Hands-on #5: □□□□□□□□

□□□□□□□

```
17
18          ②
19          handler = _lambda.Function(
20              self, 'LambdaHandler',
21              runtime=_lambda.Runtime.PYTHON_3_7,
22              code=_lambda.Code.from_inline(FUNC),
23              handler="index.handler",
24              memory_size=128,
25              timeout=cdk.Duration.seconds(10),
26              dead_letter_queue_enabled=True,
27          )
```

① □□□□ Lambda □□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□2-5□□□□□□□□□□□□□□□□□□□□["salmon", "tuna", "squid"] □□□□□□□□□□□□□□□□□□□□□ "Welcome to Cloud Sushi. Your order is XXXX" (XXXX □□□□□□□□□□□) □□□□□□□□□□□□□□□□□□

② □□□ Lambda □ <1> □□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□

```
✔ SimpleLambda

Outputs:
SimpleLambda.FunctionName = SimpleLambda-LambdaHandler212865DC-10W5VV6HTHZZI
```

Figure 83. CDK□□□□□□□□□□□

AWS □□□□□□□□□□□□

12.1.2. Lambda □□□□□

□□□□□□□□□□ Lambda □□□□□ (invoke) □□□□□□ AWS □ API □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□ handson/serverless/lambda/invoke_one.py □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□Lambda □□□□□□□□□□ □□□□□ `XXXX` □□□□□□□□□□□□□□□□□□□□ `SimpleLambda.FunctionName = XXXX` □□□□□ XXXX □□□□□□□□□□□

```
$ python invoke_one.py XXXX
```

□□□□ `"Welcome to Cloud Sushi. Your order is salmon"` □□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□

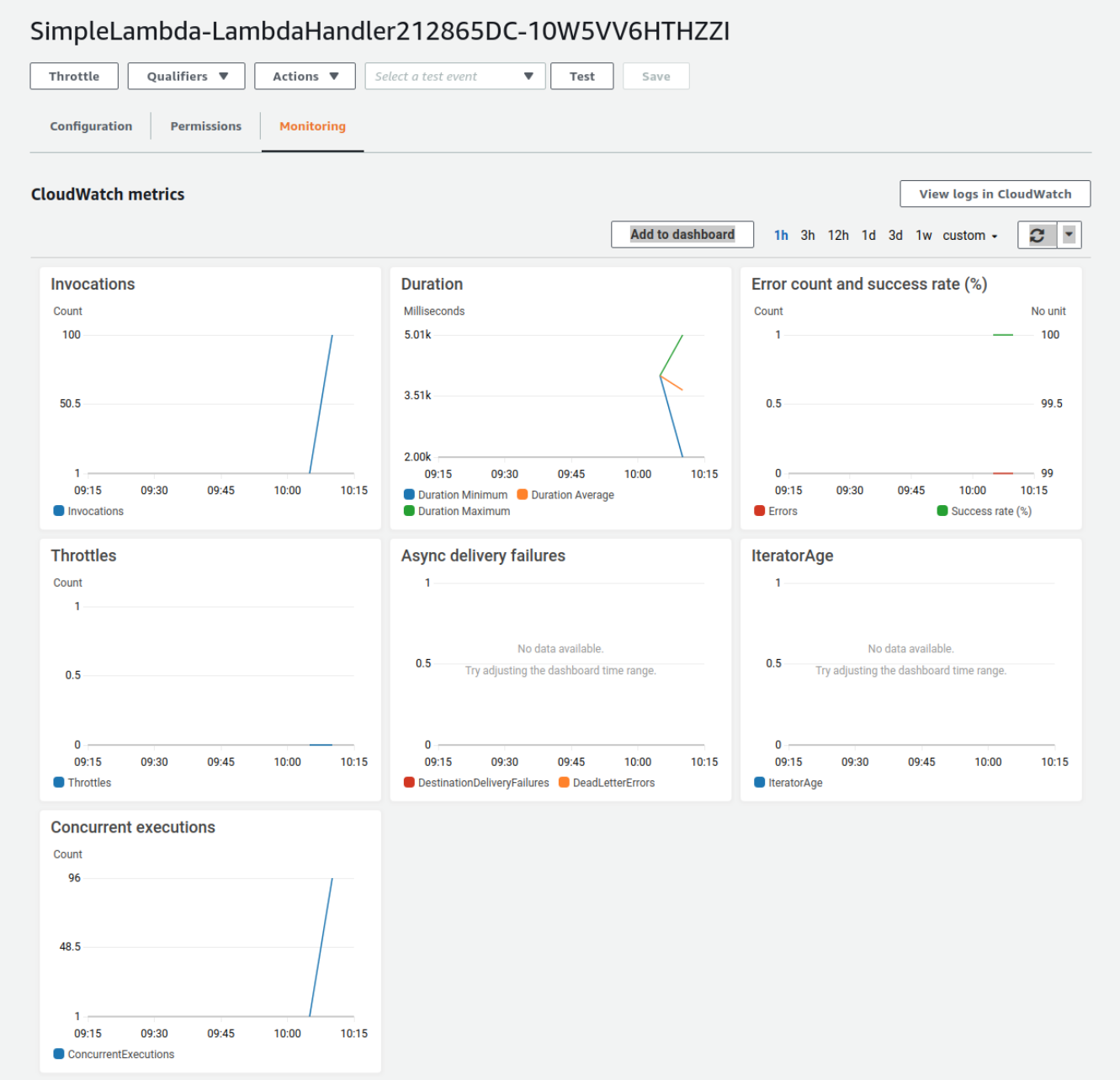


Figure 86. Lambda □□□□□ - □□□□□□□□□□□□

Figure 86 □□□□□□□□□□□□

12.1.3. □□□□□□□□

□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

```
$ cdk destroy
```

12.2. DynamoDB □□□□□□

□□□□□ DynamoDB □□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□ GitHub □ /handson/serverless/dynamodb □□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ Figure 87 □□□□□ STEP 1 □□□AWS CDK □□□□□□ DynamoDB □□□□□□□□□□□□□□□□□□□□□□□□□□□□□ STEP 2 □□□ API □□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□ DynamoDB □□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□□□

① `partition_key`: □□□□ DynamoDB □□□□□□ Partition key □□□□□□□□□□□□□□□□□ Partition key □□□□□□□□□□□□□ (□□□□) □□□□□□□□□□□□ID□□□□□□□□ □□□ Partition key □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ (□: Sort Key □□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□ "Core Components of Amazon DynamoDB" □□)□ □□□ Partition key □□ □□□□□ Partition key □ `item_id` □□□□□□□□□□□□□□

② `billing_mode`: `ddb.BillingMode.PAY_PER_REQUEST` □□□□□□□□□□ On-demand Capacity Mode □ DynamoDB □□□□□□□□ □□□ `PROVISIONED` □□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□ SimpleDynamoDb □□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□ Figure 90 □□□□□□□□□□□□□□□□□□□ "Items" □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

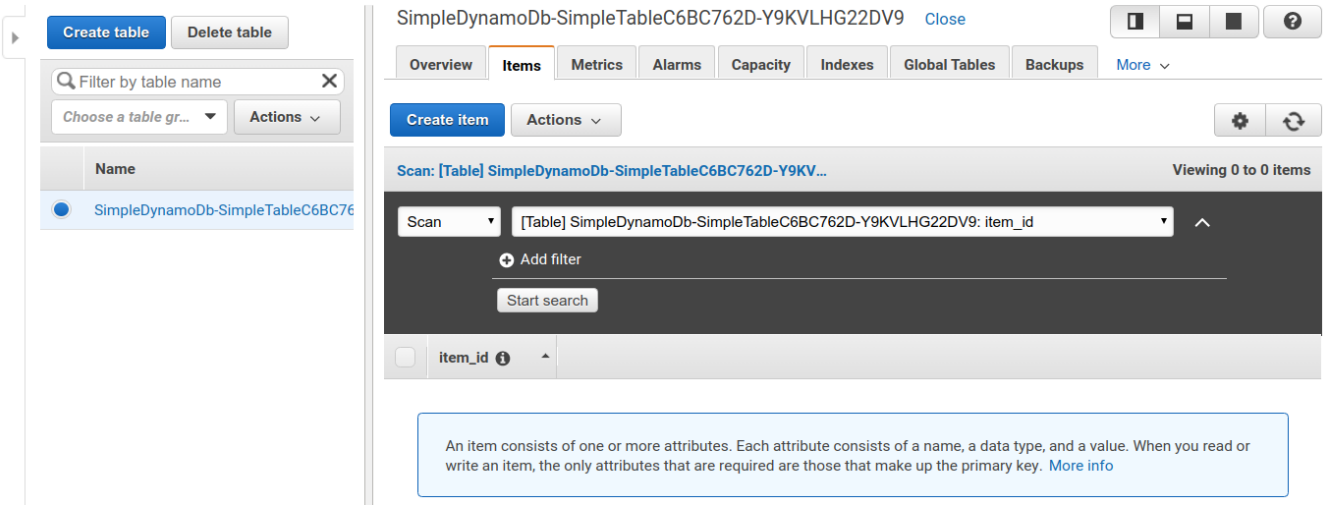


Figure 90. DynamoDB □□□□□□□ (□□□□□□□□□□)

12.2.2. □□□□□□□□□

□□□□□ Section 12.2.1 □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□ Python □ boto3 □□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□ simple_write.py □□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□

```
import boto3
from uuid import uuid4
ddb = boto3.resource('dynamodb')

def write_item(table_name):
    table = ddb.Table(table_name)
    table.put_item(
    Item={
        'item_id': str(uuid4()),
        'first_name': 'John',
        'last_name': 'Doe',
        'age': 25,
        }
    )
```

□□□□□□□□□□□□□□□□□□□□□□□□ boto3 □□□□□□□□□□□□□□□□ dynamodb □□□□□□□□□□□□□□□□□□ write_item() □□□□□ DynamoDB □□□□□□□□□□ (□□□□SimpleDynamoDb-XXXX) □□□□□□□□□□□□□□ □□□□□ put_item() □□□□□□□□□□□□□□□□□□□□□□□□ DB □□□□□□□□□□ □□□□□□□ item_id, first_name, last_name, age □4□□□□□□□□□□□□□□□□ □□□□□ item_id □□□□□□□□□□ Partition key □□□□□□□□□□ UUID4 □□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□ simple_write.py □□□□□□□□□□□□ "XXXX" □□□□□□□□□□□□□□□□□□□□□□□□□□ (SimpleDynamoDb □□□□□□□□□) □□□□□□□□□□□□□□□□□□□□□□□□□□□□

```
$ python simple_write.py XXXX
```

□□□□□□□□□□□□□□□□□ AWS □□□□□□□□□□□□□□□□ Figure 90 □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□ Figure 91 □□□□□□□□□□□□□□□□□□□□□□□□□□□□

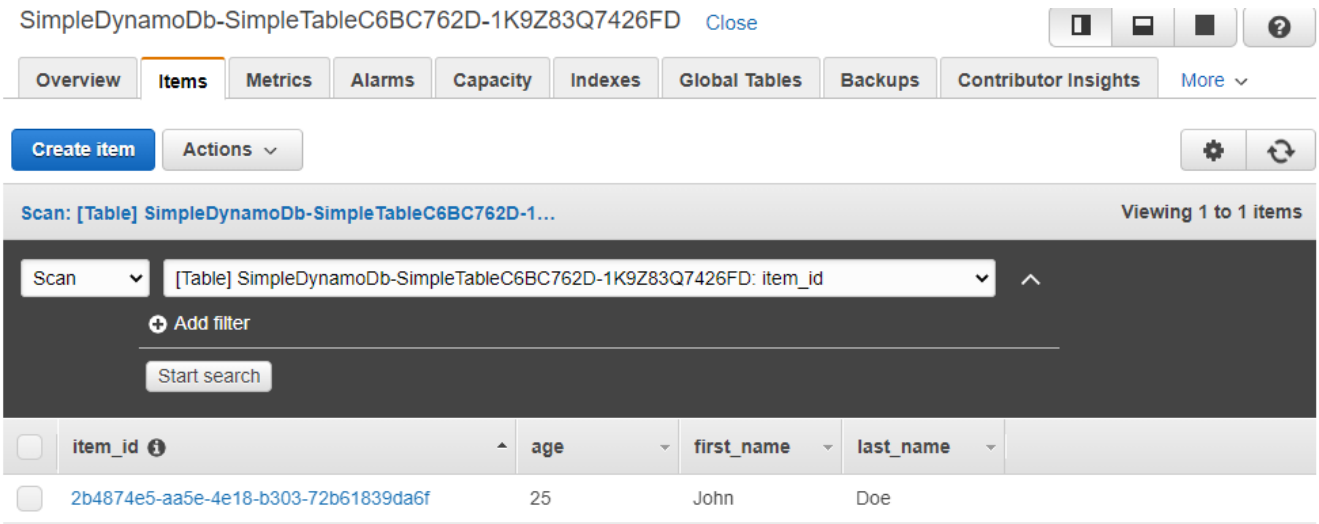


Figure 91. DynamoDB □□□□□□□□□□□□□□□□□□□□

boto3 □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□ simple_read.py □□□□□□□□

```
import boto3
ddb = boto3.resource('dynamodb')

def scan_table(table_name):
    table = ddb.Table(table_name)
    items = table.scan().get("Items")
    print(items)
```

`table.scan().get("Items")` □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□ ("XXXX" □□□□□□□□□□□□□□□□□□□□□□□□□□

```
$ python simple_read.py XXXX
```

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

12.2.3. □□□□□□□□□□□□□

DynamoDB □□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□ age □□□□□□□1□□50□□□□□□□□□□□□□□□□□□□□□□ age □2□□□□□□□□□□□□□□□□□□□

```
10          )
```

s3.Bucket() □□□□

```
$ python simple_s3.py XXXX upload tmp.txt
```

simple_s3.py □□□□□□□□□□□□□□□□□□□□□□□□□□□□

```
1 def upload_file(bucket_name, filename, key=None):
2     bucket = s3.Bucket(bucket_name)
3
4     if key is None:
5         key = os.path.basename(filename)
6
7     bucket.upload_file(filename, key)
```

bucket = s3.Bucket(bucket_name) □□□ Bucket() □□□□□□□□□□□□□□□ □□□□ upload_file() □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

S3 □□□□□□□□□□□□□□□□□□□□□□□□ **Key** □□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□ (Path) □□□□□□□□□□□□□□□□□□□□□□□□ Key □□□□□□□□□□□□□□□□ Key □□□□□□□□□ S3 □ Object storage □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ --key □□□□□□□□□□□□□ simple_s3.py □□□□□□□□□□ Key □□□□□□□□□□□□□□□□□□□□□□□□□□□□□

```
$ python simple_s3.py XXXX upload tmp.txt --key a/b/tmp.txt
```

□□□□□□□□□□□□□□□□□□□□□ a/b/tmp.txt □□□ Key □□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□ AWS □□□□□□□□□ S3 □□□□□□□□□□□□□□ S3 □□□□□□□□□□□□□□□□□□□□

□□□

Chapter 13. Hands-on #6: Bashoutter

□□□

13.2. □□□□□□□□□□□□

13.2.1. API

□□□□□□□□□□□

- □□□□□□□□□□ API □□□□□□□□ **API Gateway** (□□)□□□□□□□□□ API □ URI □□□□□□□ Lambda □□□□□□□□□□
- □□□□□ API □□□ (□□□□) □□□□□□□□ Lambda □□□□□□□
- □□□□□ (□□□□□□□□□□□□□) □□□□□□□□□□□□□□□□ (DynamoDB) □□□□□□□
- □ Lambda □□□□□ DynamoDB □□□□□□□□□□□□□□□
- □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□□□ S3 □□□□□□□□□□□□□□□□□□□□□□□ S3 □□□□□□□□□□□□□□□□□ HTML/CSS/JS □□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□ (handson/bashoutter/app.py)□

```
class Bashoutter(Stack):

    def __init__(self, scope: Construct, construct_id: str, **kwargs) -> None:
        super().__init__(scope, construct_id, **kwargs)

        ①
        # dynamoDB table to store haiku
        table = ddb.Table(
            self, "Bashoutter-Table",
            partition_key=ddb.Attribute(
                name="item_id",
                type=ddb.AttributeType.STRING
            ),
            billing_mode=ddb.BillingMode.PAY_PER_REQUEST,
            removal_policy=cdk.RemovalPolicy.DESTROY
        )

        ②
        bucket = s3.Bucket(
            self, "Bashoutter-Bucket",
            website_index_document="index.html",
            public_read_access=True,
            removal_policy=cdk.RemovalPolicy.DESTROY
        )

        common_params = {
            "runtime": _lambda.Runtime.PYTHON_3_7,
            "environment": {
                "TABLE_NAME": table.table_name
            }
        }

        ③
        # define Lambda functions
        get_haiku_lambda = _lambda.Function(
            self, "GetHaiku",
            code=_lambda.Code.from_asset("api"),
            handler="api.get_haiku",
            memory_size=512,
```

```
40            **common_params,
41        )
42        post_haiku_lambda = _lambda.Function(
43            self, "PostHaiku",
44            code=_lambda.Code.from_asset("api"),
45            handler="api.post_haiku",
46            **common_params,
47        )
48        patch_haiku_lambda = _lambda.Function(
49            self, "PatchHaiku",
50            code=_lambda.Code.from_asset("api"),
51            handler="api.patch_haiku",
52            **common_params,
53        )
54        delete_haiku_lambda = _lambda.Function(
55            self, "DeleteHaiku",
56            code=_lambda.Code.from_asset("api"),
57            handler="api.delete_haiku",
58            **common_params,
59        )
60
61        ④
62        # grant permissions
63        table.grant_read_data(get_haiku_lambda)
64        table.grant_read_write_data(post_haiku_lambda)
65        table.grant_read_write_data(patch_haiku_lambda)
66        table.grant_read_write_data(delete_haiku_lambda)
67
68        ⑤
69        # define API Gateway
70        api = apigw.RestApi(
71            self, "BashoutterApi",
72            default_cors_preflight_options=apigw.CorsOptions(
73                allow_origins=apigw.Cors.ALL_ORIGINS,
74                allow_methods=apigw.Cors.ALL_METHODS,
75            )
76        )
77
78        haiku = api.root.add_resource("haiku")
79        haiku.add_method(
80            "GET",
81            apigw.LambdaIntegration(get_haiku_lambda)
82        )
83        haiku.add_method(
84            "POST",
85            apigw.LambdaIntegration(post_haiku_lambda)
86        )
87
88        haiku_item_id = haiku.add_resource("{item_id}")
89        haiku_item_id.add_method(
90            "PATCH",
```

```
            apigw.LambdaIntegration(patch_haiku_lambda)
        )
        haiku_item_id.add_method(
            "DELETE",
            apigw.LambdaIntegration(delete_haiku_lambda)
        )
```

① □□□□□□□□□□□□□□□□□□□□□□ DynamoDB □□□□□□□□□□□□□

② □□□□□□□□□□□□□□□□□□ S3 □□□□□□□□□□□□□□

③ □□□□□ API □□□□□□ Lambda □□□□□□□□□□□ □□□ Python3.7 □□□□□□□□□□□□□ handson/bashoutter/api/api.py □□□□

④ <3> □□□□□□ Lambda □□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□ AWS □□□□□□□□□□ URL □□□□□□□□□□ DNS □□□□□□□□□□□□□□□

13.2.4. API □□□□□□□□

API □□□□□□□□□□□□□□□□□□□□

DynamoDB □ scan() □□□□□□□□□ 1MB □□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□ 1MB □□□□□□□□□□□□□□□□□□□□ scan() □□□□□□□□□□□□□□ □□□□ boto3 □□□□□□□□□□□ □□□□

13.2.5. AWS □□□□□□□□□□ (IAM)

□□□□□□□□□□□□□□□□□□□□

```
table.grant_read_data(get_haiku_lambda)
table.grant_read_write_data(post_haiku_lambda)
table.grant_read_write_data(patch_haiku_lambda)
table.grant_read_write_data(delete_haiku_lambda)
```

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ AWS □□ IAM (Identity and Access Management) □□□□□□□□□□□□□□ IAM □□□□□□□□□□□□□□□□□□□□

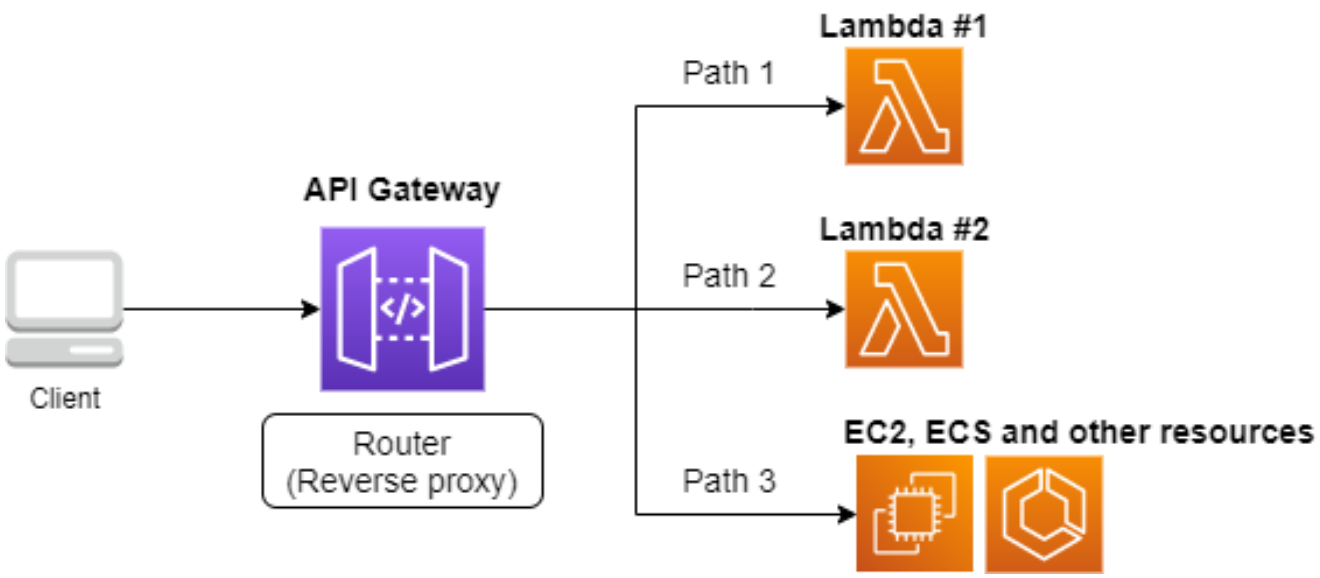


Figure 97. API Gateway

API Gateway □□□□□□□□□□□ (1□□□□□□□□□□□) □ API □□□□□□□□□□□□

```
20 )
21
22 ④
23 haiku_item_id = haiku.add_resource("{item_id}")
24 ⑤
25 haiku_item_id.add_method(
26     "PATCH",
27     apigw.LambdaIntegration(patch_haiku_lambda)
28 )
29 haiku_item_id.add_method(
30     "DELETE",
31     apigw.LambdaIntegration(delete_haiku_lambda)
32 )
```

① □□□□ `api = apigw.RestApi()` □□□□□□ API Gateway □□□□□□□□

② □□□ `api.root.add_resource()` □□□□□□□□□□□□ `/haiku` □□□ API □□□□□□□□□□

③ □□□□ `add_method()` □□□□□□□ `GET`, `POST` □□□□□□ `/haiku` □□□□□□□□□□□

④ □□□□ `haiku.add_resource("{item_id}")` □□□□ `/haiku/{item_id}` □□□ API □□□□□□□□□□

⑤ □□□□ `add_method()` □□□□□□□□□ `PATCH`, `DELETE` □□□□□□ `/haiku/{item_id}` □□□□□□□□□□□

□□□□□□ API Gateway □□□□□□□□□□□□□□□□□□□ API □□□□□□□□□□□□□□□□□Lambda □□□□□□□□□□□□□□

□□□□□□□□ □□ API □□□□□□□ □□□□□ URL □□□ API □□□□□□□□□□□□□□□□□□□□ □□□□ `api.example.com` □□□□□□□□□□□□□□□□□□□□□□□□ AWS □□□□□□□□□ URL □□□□□□□□□ DNS □□□□□□□□□□□□□□□

API Gateway □□□ API □□□□□□□□□ `default_cors_preflight_options=` □□□□□□□□□□ Cross Origin Resource Sharing (CORS) □□□□□□□□□□□ □□□□□□□□□□□□ Web □□□□□□□□□□ API □□□□□□□□□□□□□□□□□□□

13.3. □□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□

`XXXX`, `Bashoutter.BucketUrl = YYYY` □□□□□□□□□□□□□□□□□□□□□□□□

```
✔ Bashoutter

Outputs:
Bashoutter.BashoutterApiEndpointE9A7B94F = https://ig5181x0bd.execute-api.ap-northeast-1.amazonaws.com/prod/
Bashoutter.BucketUrl = bashoutter-bashoutterbucket6f28d9f3-zmi05o477p6r.s3-website-ap-northeast-1.amazonaws.com
```

Figure 98. CDK□□□□□□□□□□□

AWS □□□□□□□□□□□□□□□□□□□□

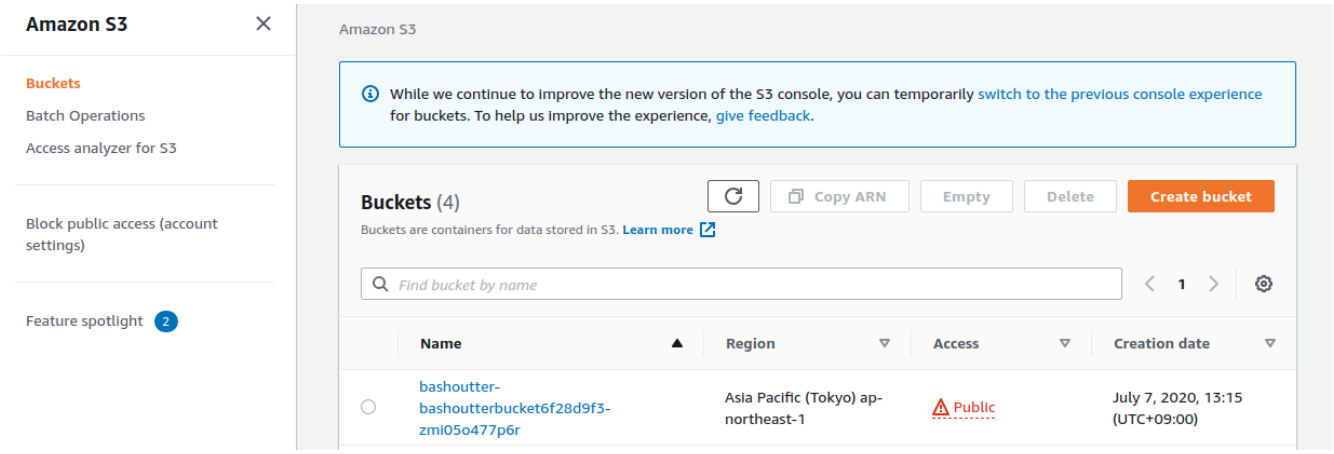


Figure 101. S3 □□□□□□□□

□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ API □□□□□□□□□□ URL (Bashoutter.BashoutterApiEndpoint = XXXX □□□□□□ XXXX □□□□□) □□□□□□□□□□□□□□□□□□□□□□□□

```
$ export ENDPOINT_URL=XXXX
```

□□□□□□□□□□□□□□□□□□□□ GET /haiku □ API □□□□□□□□□□□

```
$ http GET "${ENDPOINT_URL}/haiku"
```

□□□□□□□□□□□□□□□□□□□

```
        "third": "□□□",
        "username": "□□□□"
    }
]
```

□□□□□□□

□□□ `PATCH /haiku/{item_id}` □□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□ `item_id` □□□□□□□□□□ `XXXX` □□□□□□□□□□□□□□□□

```
$ http PATCH "${ENDPOINT_URL}/haiku/XXXX"
```

`{"description": "OK"}` □□□□□□□□□□□□□□□□□□ □□ GET □□□□□□□□□□□□□□□□□□ (`likes`) □1□□□□□□□□□□□□□□□

```
$ http GET "${ENDPOINT_URL}/haiku"
...
[
    {
        ...
        "likes": 1.0,
        ...
    }
]
```

□□□□ DELETE □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ `XXXX` □ `item_id` □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

```
$ http DELETE "${ENDPOINT_URL}/haiku/XXXX"
```

□□ GET □□□□□□□□□□□□□□□□□□□□ (`[]`) □□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□ API □□□□□□□□□□□□□□□□□□□□

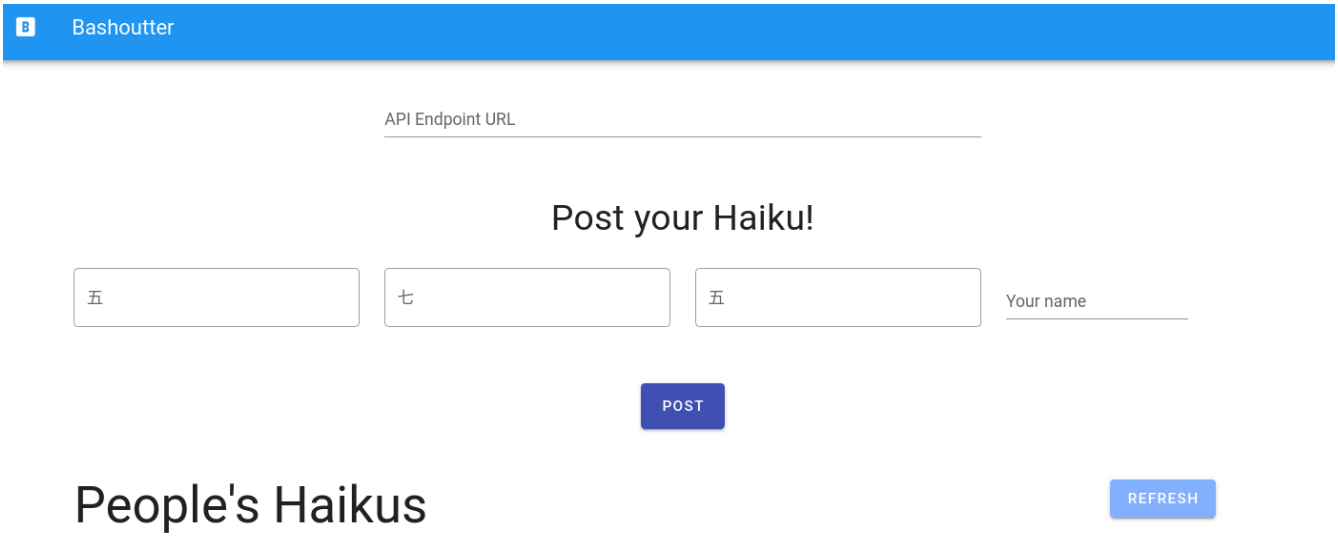


Figure 103. "Bashoutter" □ GUI □□

□□□□□□□□□□□□□□□□ "API Endpoint URL" □□□□□□□□□□□□□□□□□□□□□□□□□□□□ **API Gateway □ URL** □□□□□ (□□□□□□□□□□□□□□□□API Gateway □ URL □□□□□□□□□□□□□□□□□□□□□□□□□□□□ GUI □□□□□□□□□□□)□ □□□□□□□□□□ "REFRESH" □□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□

13.8. □□

□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□

Chapter 14. □□□

Chapter 15. Appendix: □□□□

□□□

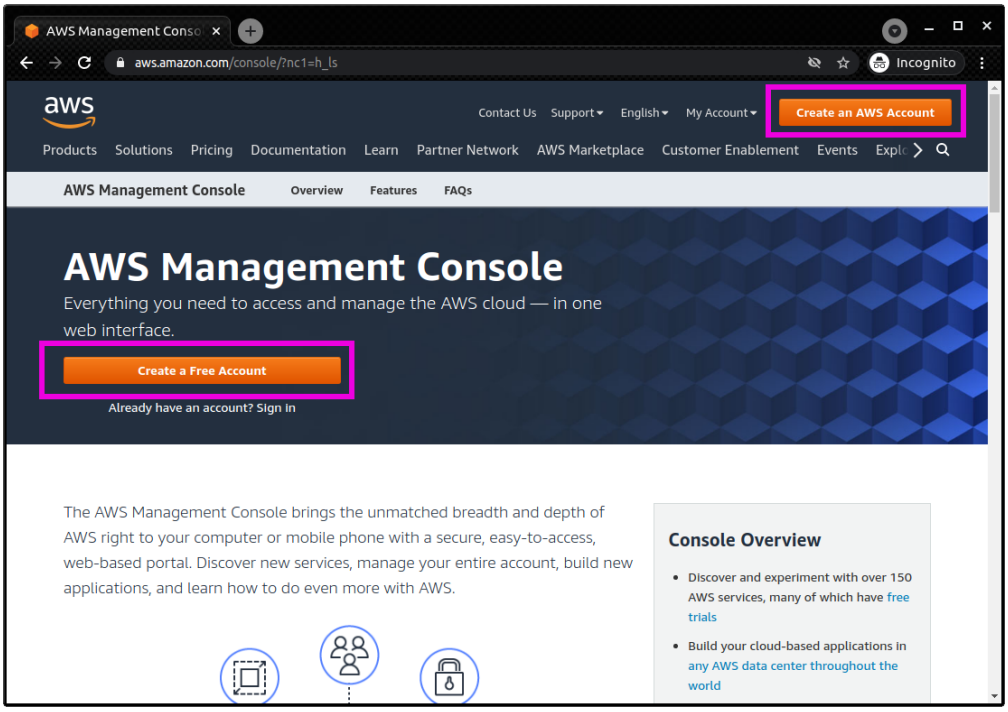


Figure 104. □□□□□□ *(1): AWS* □□□□□□□□□□□

□□□□□□□□□□

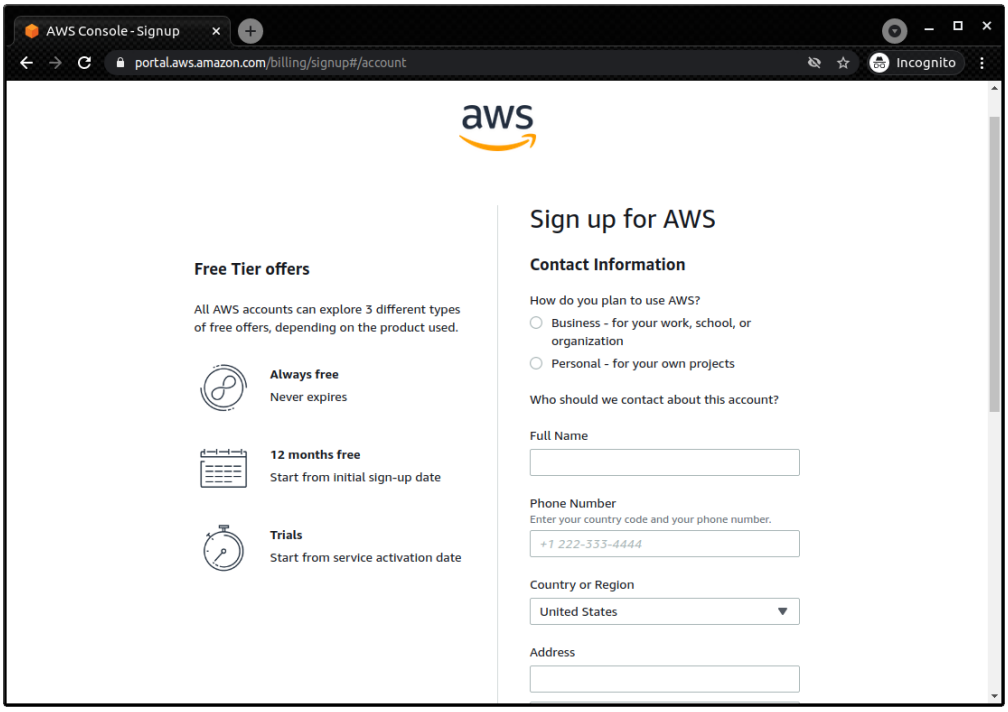


Figure 106. □□□□□□ *(3):* □□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ (Figure 107)□ □□□ AWS □□□□□□□□□□□□□□□□□□□□

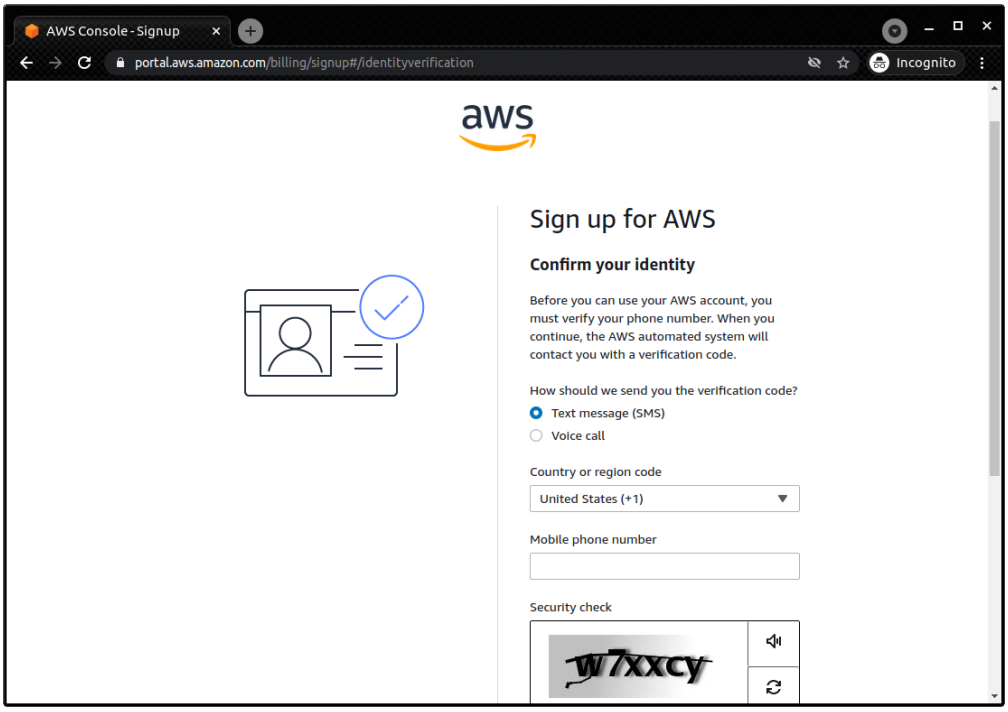

AWS Console - Signup
portal.aws.amazon.com/billing/signup#/identityverification
Incognito
aws
Sign up for AWS
Confirm your identity
Before you can use your AWS account, you must verify your phone number. When you continue, the AWS automated system will contact you with a verification code.
How should we send you the verification code?
Text message (SMS)
Voice call
Country or region code
United States (+1)
Mobile phone number
Security check
w7xxcy

Figure 108. □□□□□□ *(5):* □□□□□□□□□□□□

□□□□□□□□□□□

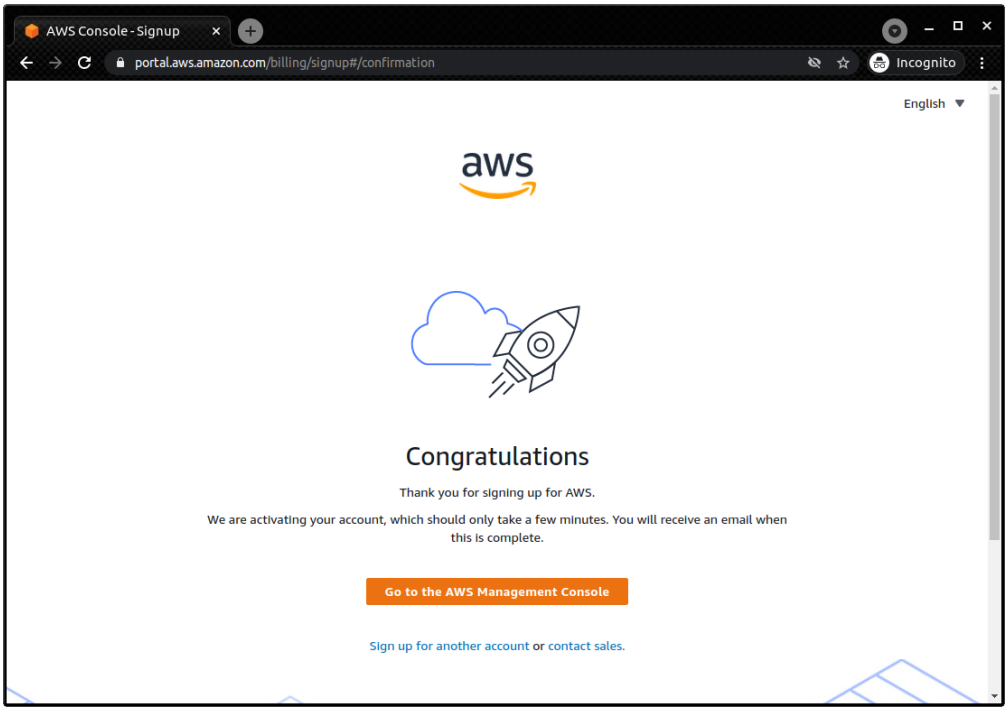


Figure 110. □□□□□□ (7): □□□□□□□□□□□□□□

15.2. AWS □□□□□□□□□□□□□

AWS □□□□□□□□□□□ AWS CLI □ AWS CDK □□ AWS □ API □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ AWS CLI/CDK □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ AWS □□□□□□□□□□□□□ □□□□□□□□□□□□□ "Understanding and getting your AWS credentials" □□□□

1. AWS □□□□□□□□□□□□□□□
2. □□ "My Security Credentials" □□□ (Figure 111)
3. "Access keys for CLI, SDK, & API access" □□□□□ "Create accesss key" □□□□□□□□□□□□ (Figure 112)
4. □□□□□ Access key ID, Secret access key □□□□□□□□ (□□□□□□□□□□□□□□□□□□)□
5. □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
6. □□□□□□□□□□□□□□□□ `~/.aws/credentials` □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ (□□□□ Section 15.3)□

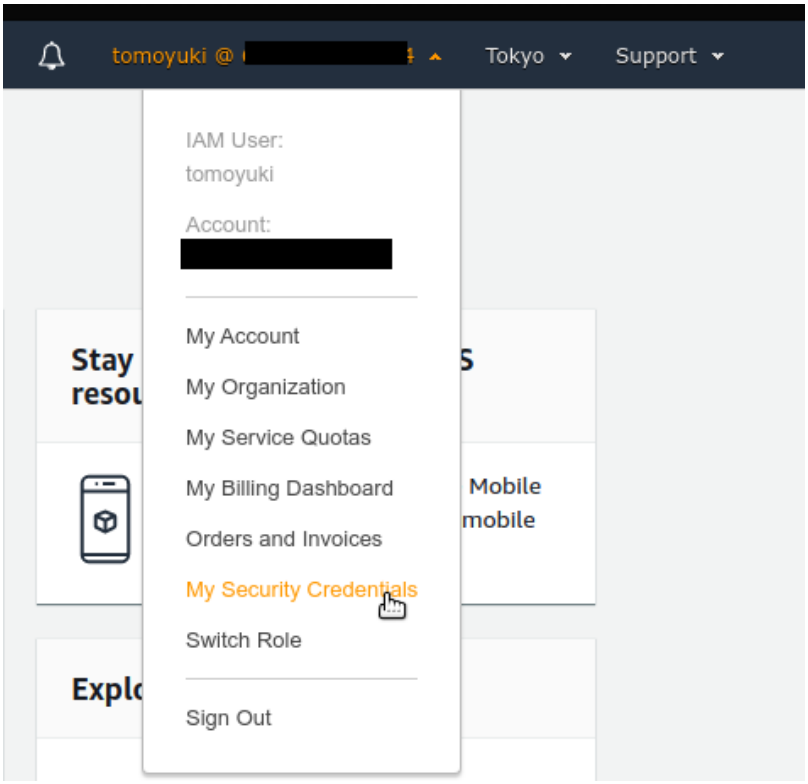


Figure 111. AWS □□□□□□□□□□□□1

Figure 112. AWS □□□□□□□□□□□□2

**AWS Educate Starter Account □□□□□□□□□□□□

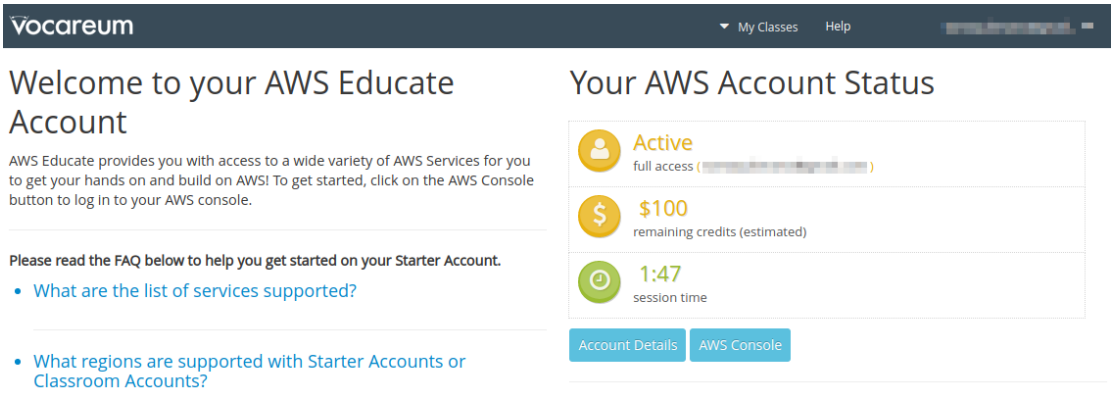


Figure 113. vocareum □□□□□

```
AWS Access
   Session started at: 2021-06-20T18:29:05-0700
   Session to end  at: 2021-06-20T21:29:05-0700
   Remaining session time: 2h18m12s

AWS Starter account
   Term: 364 days 23:13:23

AWS CLI:
   Copy and paste the following into ~/.aws/credentials
```

```
[default]
aws_access_key_id=A[redacted]
aws_secret_access_key=[redacted]
aws_session_token=[redacted]
```

Figure 114. vocareum □□ AWS □□□□□□□□□□□□

15.3. AWS CLI □□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□ (Linux □□) □□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□

```
$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
$ unzip awscliv2.zip
$ sudo ./aws/install
```

□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□ ~/.aws/credentials □ ~/.aws/config□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□ cat □□□□□□□□□□□□□□□□□□□□□□□□

```
$ cat ~/.aws/credentials
[default]
aws_access_key_id = XXXXXXXXXXXXXXXXXX
aws_secret_access_key = YYYYYYYYYYYYYYYYYY

$ cat ~/.aws/config
[profile default]
region = ap-northeast-1
output = json
```

~/.aws/credentials □□□□□□□□□□□ ~/.aws/config □□ AWS CLI □□□□□□□□□□□□□

□□□□□□□□□ [default] □□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□□ default □□□□□□□□□□□□ [myprofile] □□□□□□□□□□□□□□□□□□□□□□□□□□□□

AWS CLI □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

```
$ aws s3 ls --profile myprofile
```

□□□□□ --profile □□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□ --profile □□□□□□□□□□□□□□□□□□□□□□□□□□□□ AWS_PROFILE □□□□□□□□□□□□□□□□□□□

```
$ export AWS_PROFILE=myprofile
```

□□

```
export AWS_ACCESS_KEY_ID=XXXXXX
export AWS_SECRET_ACCESS_KEY=YYYYYY
export AWS_DEFAULT_REGION=ap-northeast-1
```

□□□□□□□□□□□□ ~/.aws/credentials □□□□□□□□□□□□□□□□□□□□

```
$ sudo npm install -g aws-cdk
```

□□□□□□□□□AWS CDK version 1.100.0 □□□□□□ CDK □□□□□□□□□□

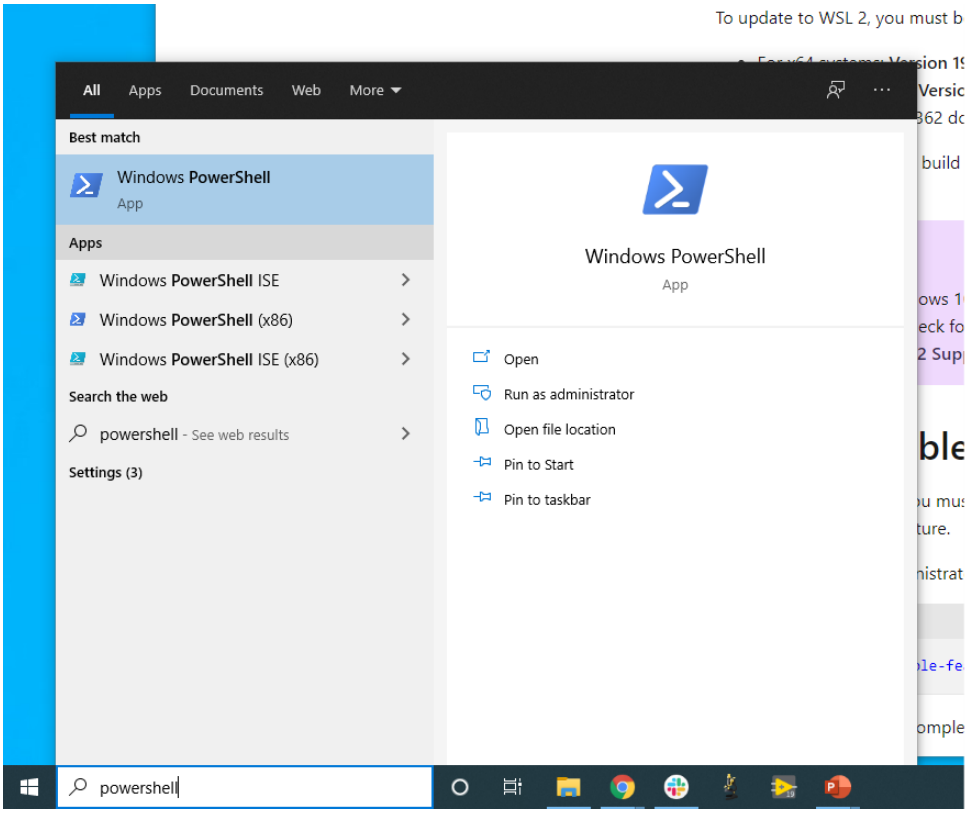


Figure 115. □□□□□□□□ *PowerShell* □□□

PowerShell □□□□□□□□□□□□□□□□□□□□□□

```
dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all
/norestart
```

□□□□□"The operation completed successfully." □□□□□□□□□□□□□□ □□□ WSL □ enable □□□□

□□□□□□□□□□ Administrator □□□□□□ PowerShell □□□□□□□□□□□□□□□

```
dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart
```

□□□□□ "The operation completed successfully." □□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□ Linux kernel update package □□□□□□□□□□□□□□□□□□□ https://wslstorestorage.blob.core.windows.net/wslblob/wsl_update_x64.msi

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□ PowerShell □□□□□□□□□□□□□□□□□

```
wsl --set-default-version 2
```

□□□□□□□□□□□□ Linux distribution □□□□□□□□□□□ □□□□□ Ubuntu 20.04 □□□□□□□□□□□□

Microsoft store □□□□□□□□□□□□□□□□ Ubuntu □□□□□□□ Ubuntu 20.04 LTS □□□□□□□□□□□□□□□□□□□□□□□□□□ "Get"

□□□□□□□□□□□ (Figure 116)□ □□□□□□□□ Ubuntu 20.04 □□□□□□□□□□□□□□

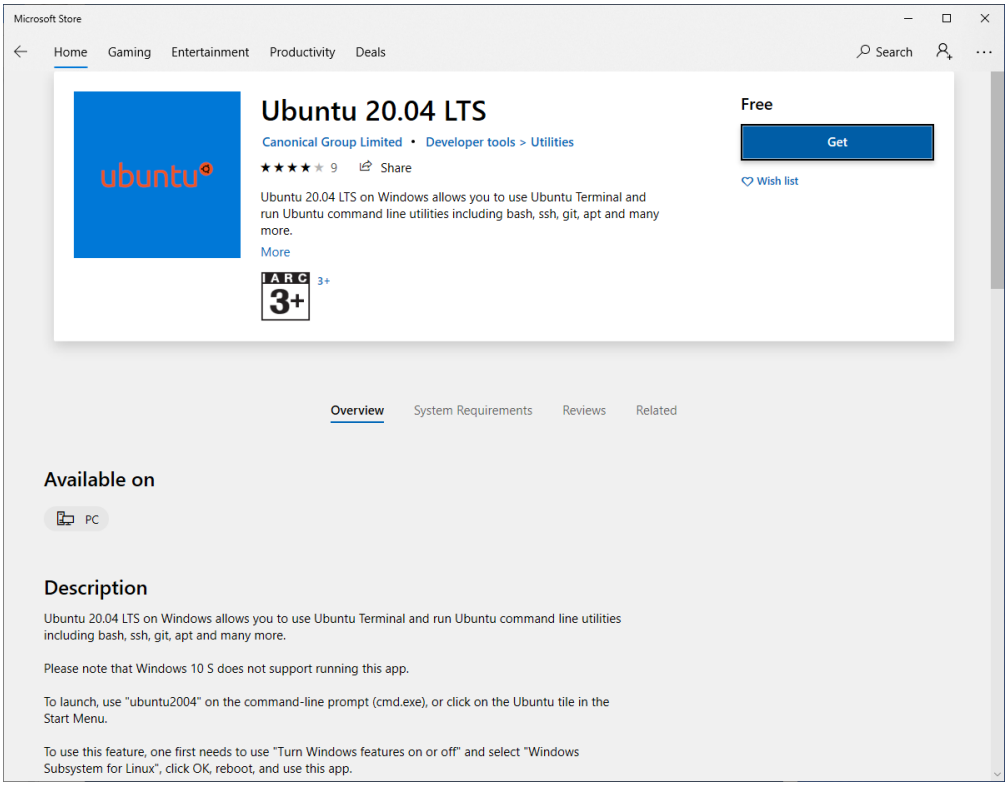


Figure 116. Microsoft store □□ Ubuntu 20.04 □□□□□□□□

Ubuntu 20.04 □□□□□□□□□□□□□□

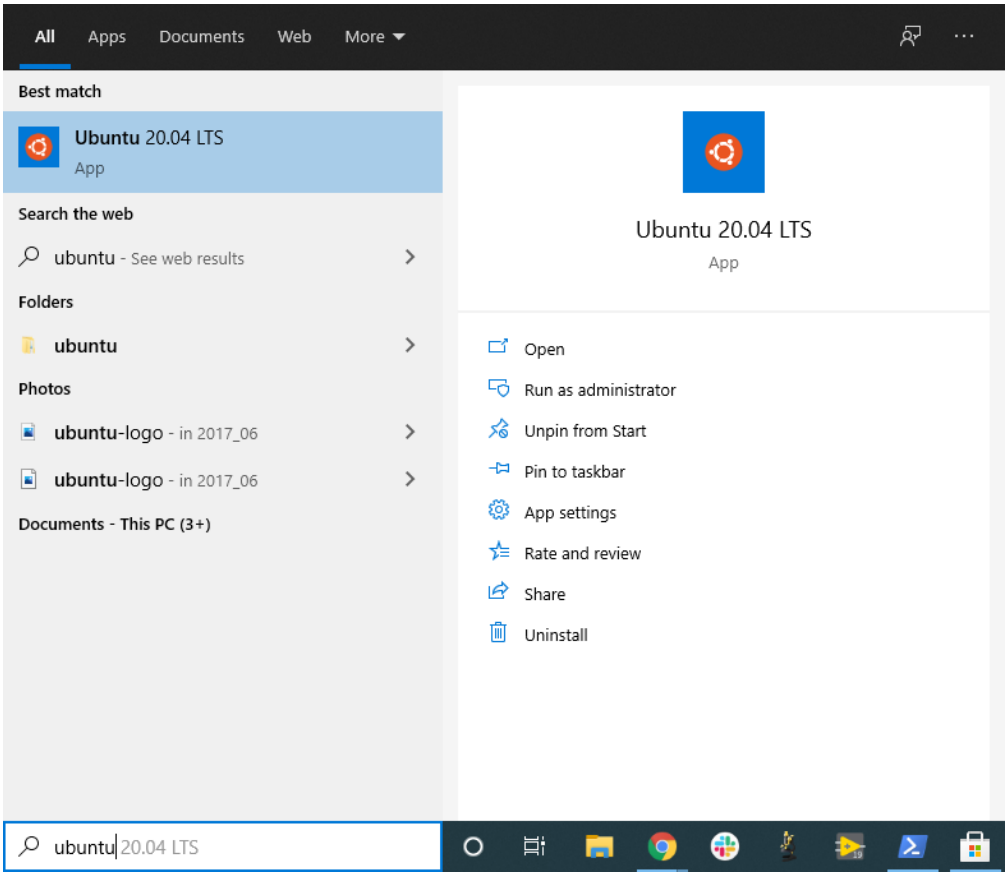


Figure 117. Ubuntu 20.04 □□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□ (Figure 118)□ `ls`, `top` □□□□□□□□□□□□□□□□ WSL □□□□□□□□□□□□□□□□□□□□□□□□

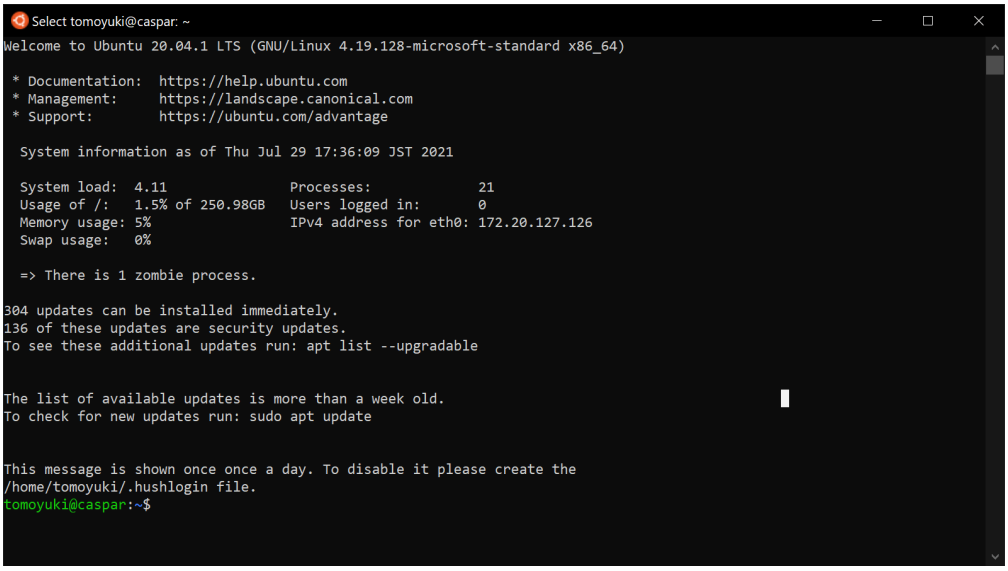


Figure 118. WSL □□□□□

□□□□□□□□□□ Windows Terminal □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ WSL □□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

15.6. Docker □□□□□□□□

Docker □□□□□□□□□□□□ OS □□□□□□□□□

Mac □□□□□□□ Docker Desktop □□□□□□□□□□□□ □□□□□□□□□□□□□ Docker □□□□□□□□ □□□ Mac □□ Docker Desktop

□□□

```
$ python -m venv .env
```

□□□□□□□ □□□□□□ .env/ □□□□□□□□

□□□

Chapter 16. □□

□□□

Chapter 17. □□□□

□□ □□ (Tomoyuki Mano)

□□□□□□□ (□□□□□□□

Chapter 18. □□□□□

□□□□□□□□□□□□□□□□□□□□□□□□ CC BY-NC-ND 4.0 □□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□