IOWA

CS5630

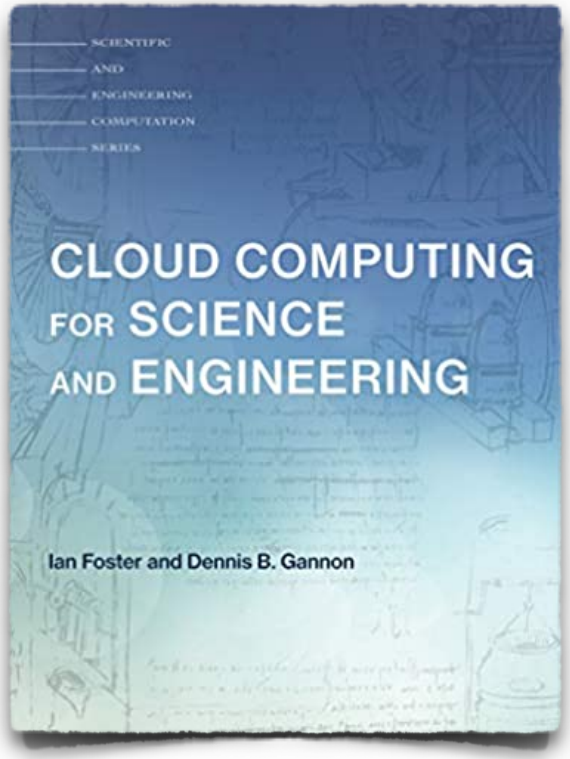
Foundation (3): Compute in the Cloud

Prof. Supreeth Shastri
Computer Science
The University of Iowa

Lecture goals

Technical introduction to computing resources on the cloud

- ***Infrastructure as a Service (IaaS)***
- *Platform as a Service (PaaS)*
- *Software as a Service (SaaS)*
- *Other emerging models*



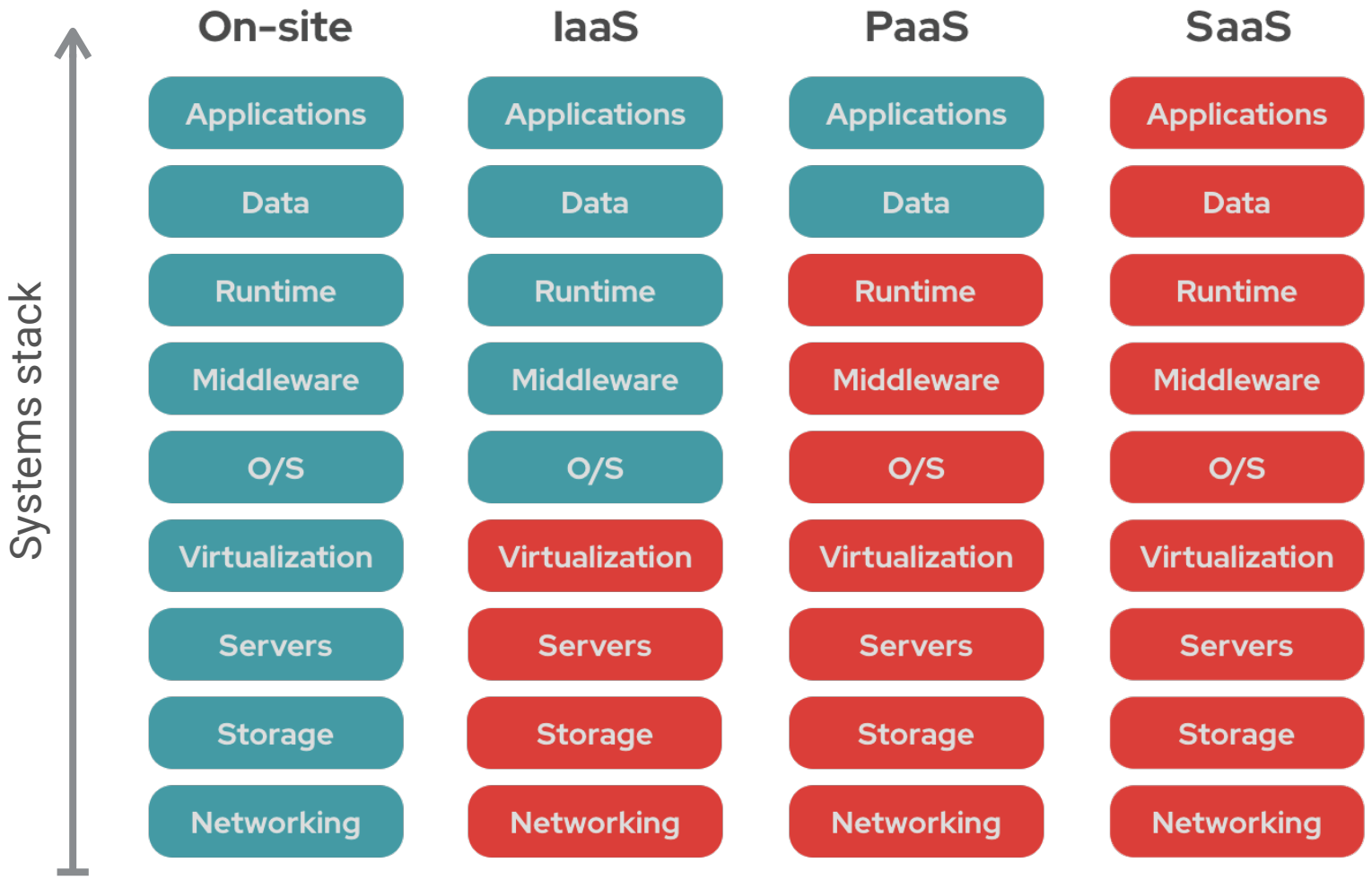
Chapters 4, 5

aws

Google Cloud

IOWA

Visualizing compute in the cloud



You manage

Service provider manages

Image courtesy: Redhat

IaaS Offerings are Redefining the Compute Server

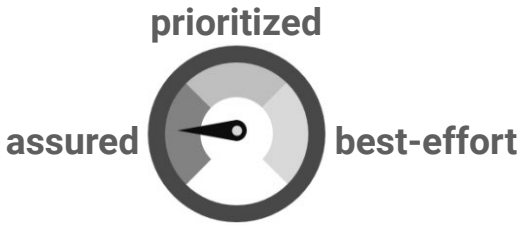
Static characteristics

Price, performance, and failure characteristics that do not change in an application's lifetime

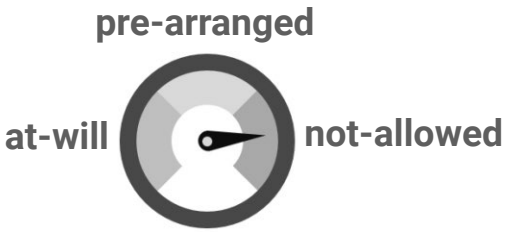
The Cloud Era

“A rose by any other name *may not* ~~would~~ smell as sweet”

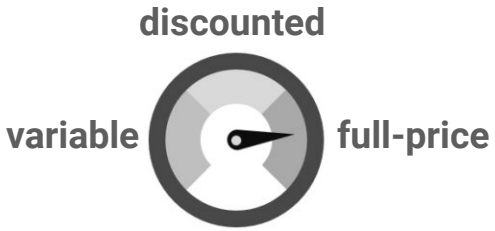
All service-level characteristics are controlled in fine granularity



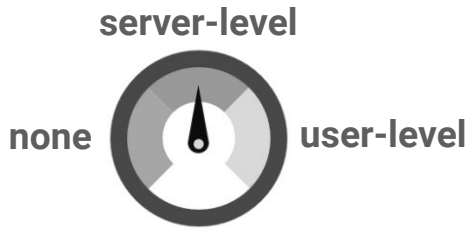
Availability



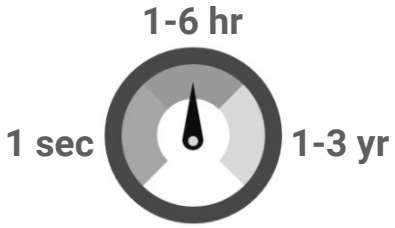
Revocability



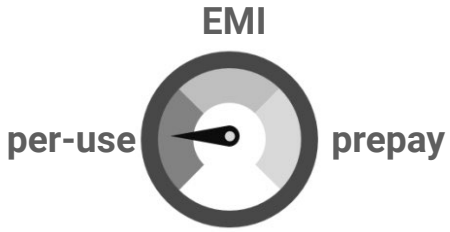
Pricing Model



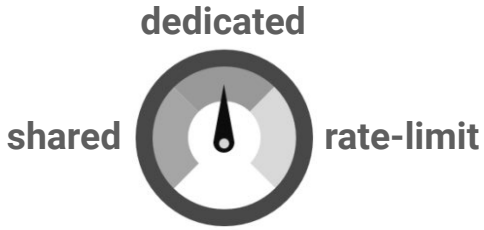
Isolation



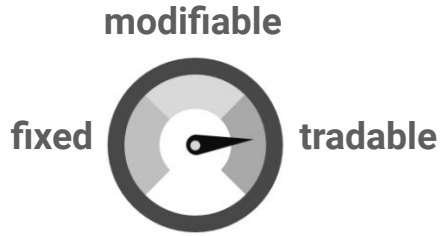
Time Commitment



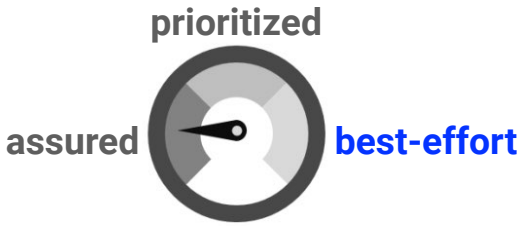
Billing Frequency



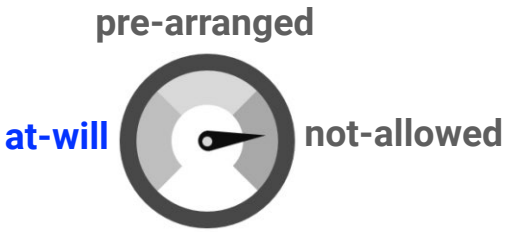
Performance Guarantees



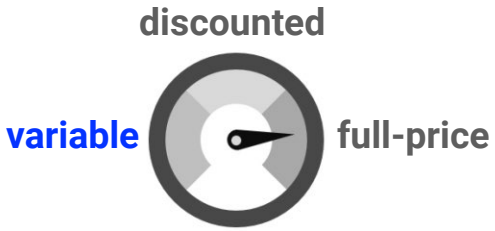
Terms of Contract



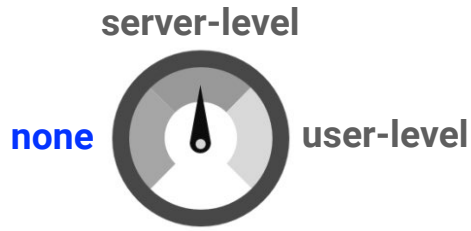
Availability



Revocability

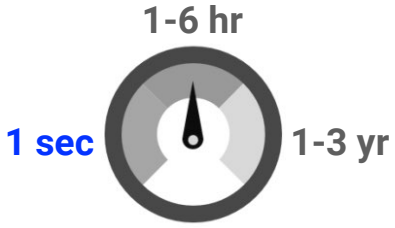


Pricing Model

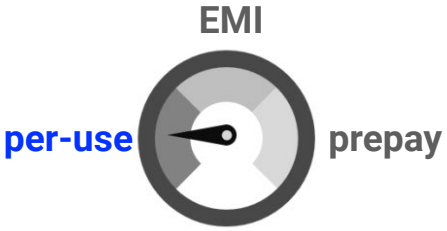


Isolation

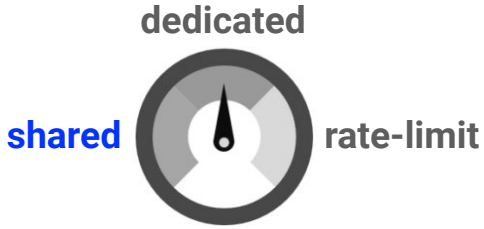
Spot Server + 15 other IaaS offerings on EC2



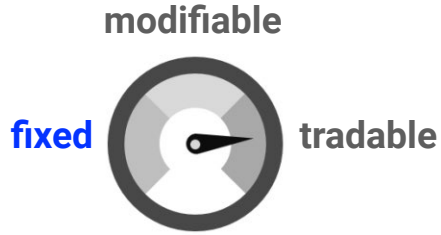
Time Commitment



Billing Frequency



Performance Guarantees



Terms of Contract

EC2 Contract Diversity

EC2 contract	Cost model	Performance	Availability	Revocation risk
On-demand	fixed cost per sec	low variability	high	no
Reserved	fixed cost per year	low variability	guaranteed	no
Spot	variable cost per sec	low variability	not guaranteed	yes (market driven)
Defined-duration	variable cost per hour	low variability	not guaranteed	yes (on expiry)
Burstable	fixed cost per sec	high variability	high	no
Dedicated	fixed cost per sec	no variability	high	no

EC2 Location Diversity

- **25 geographic regions** (2x more than the next largest provider)
- **81 availability zones** (i.e., fully isolated datacenters)
- **230+ edge** locations/caches

EC2 Hardware Spec

Five categories

- General purpose
- Compute optimized
- Memory optimized
- Storage optimized
- Accelerated

400

instance types

C6g | C6gn | C5 | C5a | C5n | C4

Model	vCPU	Memory (GiB)	Instance Storage (GiB)	Network Bandwidth (Gbps)***	EBS Bandwidth (Mbps)
c5.large	2	4	EBS-Only	Up to 10	Up to 4,750
c5.xlarge	4	8	EBS-Only	Up to 10	Up to 4,750
c5.2xlarge	8	16	EBS-Only	Up to 10	Up to 4,750
c5.4xlarge	16	32	EBS-Only	Up to 10	4,750
c5.9xlarge	36	72	EBS-Only	10	9,500
c5.12xlarge	48	96	EBS-Only	12	9,500
c5.18xlarge	72	144	EBS-Only	25	19,000
c5.24xlarge	96	192	EBS-Only	25	19,000
c5.metal	96	192	EBS-Only	25	19,000

location / hardware

Contract diversity is omnipresent in the real world

Short-term rental	Assisted living	Long-term room share	Lease-to-own

Pricing the EC2 Servers

- Price is a function of hardware spec, location, and choice of contract
- To illustrate, let's consider two instance types in two regions under five different contracts
  - *c5 series is compute-optimized and x1e series is memory optimized*
  - *us-east-1 is located in Virginia, whereas ap-northeast-3 is in Osaka*

prices decrease this way

c5.24xlarge	us-east-1	ap-northeast-3
On-demand	4.080	5.136
Dedicated	3.366	4.237
Reserved (1y)	2.570	3.236
Defined-duration	2.244	2.825
Spot	1.555	1.596

price diff. reduced

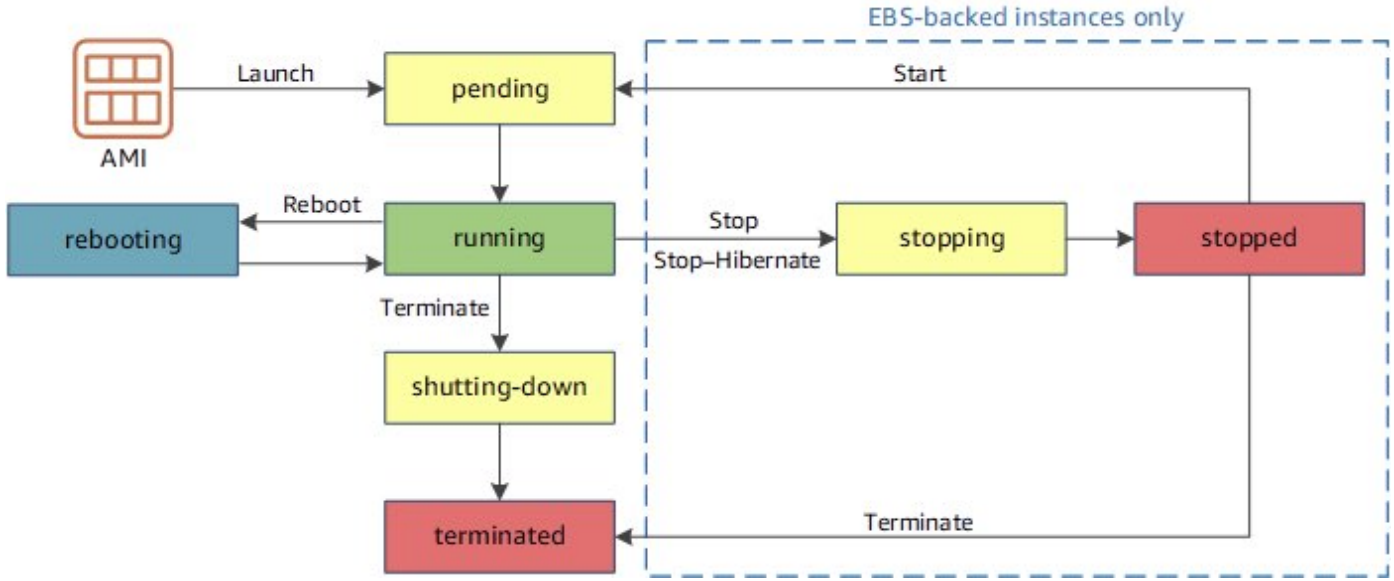
x1e.32xlarge	us-east-1	ap-northeast-3
On-demand	26.688	38.688
Dedicated	29.357	42.557
Reserved (1y)	16.446	23.839
Defined-duration	14.678	21.278
Spot	8.006	N/A

price rise

unavailability

Working with VM on EC2

EC2 instance lifecycle



- This diagram should be familiar from your OS class (process lifecycle, VM lifecycle etc)
- All instances will be in one of the seven states; but billed only in the <running> state
- Hibernation (shown in blue box) is only possible when the root file system is backed by EBS
- Instance lifecycle can be managed via a web console, CLIs, or SDK-supported APIs
  - *For this exercise, we will use AWS CLIs*
  - *AWS CLI is an open-source tool that allows full management via command-line shell of Linux/MacOS/Windows*

Image courtesy: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-lifecycle.html

1. Create a key pair
2. Set up a security group
3. Launch instance
4. Associate public IP
5. Connect to instance
6. Manage lifecycle

```
$ aws ec2 create-key-pair --key-name cs5630 --output text > cs5630.pem
```

```
——BEGIN RSA PRIVATE KEY-----
EXAMPLEKEYKCAQEAy7WZhaDsrA1W3mRlQtvhwyORRX8gnxgDAfRt/gx42kWXsT4rXE/b5CpSgie/
vBoU7jLxx92pNHoFnByP+Dc21eyyz6CvjTmWA0JwfWiW5/akH7iO5dSrvC7dQkW2duV5QuUdE0QW
Z/aNxMniGQE6XAgfwlnXVBwrerrQo+ZWQeqiUwwMkuEbLeJFLhMCvYURpUMSC1oehm449ilx9X1F
G50TCFeOzfl8dqqCP6GzbPaIjiU19xX/azOR9V+tpUOzEL+wmXnZt3/nHPQ5xvD2OJH67km6SuPW
oPzev/D8V+x4+bHthfSjR9Y7DvQFjfBVwHXigBdtZcU2/wei8D/HYwIDAQABAoIBAGZ1kaEvnrqu
/uler7vgIn5m7lN5LKw4hJLAIW6tUT/fzvtcHK0SkbQCQXuriHmQ2MQyJX/0kn2NfjLV/ufGxbL1
mb5qwMGUnEpJaZD6QSSs3kICLwWUYUiGfc0uiSbmJoap/GTLU0W5Mfcv36PaBUNy5p53V6G7hXb2
bahyWyJNfjLe4M86yd2YK3V2CmK+X/BOsShnJ36+hjrXPPWmV3N9zEmCdJjA+K15DYmhm/tJWSD9
81oGk9TopEp7CkIfatEATyyZiVqoRq6k64iuM9JkA3OzdXzMQexXVJ1TLZVEH0E7bhlY9d8O1ozR
oQs/FiZNAx2iijCWyv0lpjE73+kCgYEA9mZtyhkHkFDpwrSM1APaL8oNAbbjwEy7Z5Mqfql+lIp1
YkriL0DbLXlvRAH+yHPRit2hHOjtUNZh4Axv+cpg09qbUI3+43eEy24B7G/Uh+GTfbjsXsOxQx/x
p9otyVwc7hsQ5TA5PZb+mvkJ5OBEKzet9XcKwONBYELGhnEPe7cCgYEA06Vgov6YHleHui9kHuws
ayav0elc5zkxjF9nfHFJRry21R1trw2Vdpn+9g481URrpzWVOEihvm+xTtmaZlSp//lkq75XDwnU
WA8gkn6O3QE3fq2yN98BURsAKdJfJ5RL1HvGQvTe10HLYYXpJnEkHv+Unl2ajLivWUt5pbBrKbUC
gYBjbO+OZk0sCcpZ29sbzjYjpIddErySIyRX5gV2uNQwAjLdp9PfN295yQ+BxMBXiIycWVQiw0bH
oMo7yykABY7Ozd5wQewBQ4AdSlWSX4nGDtsiFxWiI5sKuAAeOCbTosy1s8w8fxoJ5Tz1sdoxNeGs
Arq6Wv/G16zQuAE9zK9vvwKBgF+09VI/1wJBirsDGz9whVWfFPrTkJNvJZzYt69qezxlsjgFKshy
WBhd4xHZtmCqpBPlAymEjr/TOlbxyARmXMnIOWIAnNXMGB4KGSyl1mzSVAoQ+fqR+cJ3d0dyPl1j
jjb0Ed/NY8frlNDxAVHE8BSkdsx2f6ELEyBKJSRr9snRAoGAMrTwYneXzvTskF/S5Fyu0iOegLDa
NWUH38v/nDCgEpIXD5Hn3qAEcju1IjmbwlvtW+nY2jVhv7UGd8MjwUTNGItdb6nsYqM2asrnF3qS
VRkAKKKYeGjkpUfVTrW0YFjXkfcrR/V+QFL5OndHAKJXjW7a4ejJLncTzmZSpYzwApc=
-----END RSA PRIVATE KEY-----
```

1. Create a key pair
2. Set up a security group
3. Launch instance
4. Associate public IP
5. Connect to instance
6. Manage lifecycle

```
$ aws ec2 create-security-group --group-name mySG --description “My SG”
{
    "GroupId": "sg-903004f8"
}
```

1. Create a key pair
2. Set up a security group
3. Launch instance
4. Associate public IP
5. Connect to instance
6. Manage lifecycle

```
$ aws ec2 run-instances --image-id <ami-173d747e> --count 1 --instance-
  type t3.micro --key-name cs5630 --security-groups mySG
{
    "OwnerId": "123456789012",
    "ReservationId": "r-5875ca20",
    "Instances": [
        {
            "PublicDnsName": “ec2-192-0-2-1.us-west—2.compute.amazonaws.com”,
            "Platform": “linux",
            "State": {
                "Code": 0,
                "Name": "pending"
            },
            "LaunchTime": "2021-09-05T02:42:39.000Z",
            "InstanceId": "i-5203422c",
            "ImageId": "ami-173d747e",
            "KeyName": "cs5630",
            "InstanceType": "t3.micro",
            "Placement": {
               "AvailabilityZone": "us-west-2b"
            },
           "BlockDeviceMappings": [
                {
                    "DeviceName": "/dev/sda1",
                    "Ebs": {
                        "Status": "attached",
                        "DeleteOnTermination": true,
                        "VolumeId": "vol-877166c8",
                    }
                }
            ],
           "RootDeviceName": "/dev/sda1",
           "RootDeviceType": "ebs",
        }
    ]
}
```

1. Create a key pair
2. Set up a security group
3. Launch instance
4. Associate public IP
5. Connect to instance
6. Manage lifecycle

```
$ aws ec2 associate-address --instance-id i-5203422c --public-ip 198.51.100.0


$ aws ec2 authorize-security-group-ingress --group-name mySG --protocol tcp --
  port 22 --cidr 198.51.100.0
```

1. Create a key pair
2. Set up a security group
3. Launch instance
4. Associate public IP
5. **Connect to instance**
6. Manage lifecycle

```
$ ssh -i cs5630.pem ec2-user@198.51.100.0
$ scp -i cs5630.pem <src-file> ec2-user@198.51.100.0:/<dest-path>
```

1. Create a key pair
2. Set up a security group
3. Launch instance
4. Associate public IP
5. Connect to instance
6. Manage lifecycle

```
$ aws ec2 stop-instances --instance-ids i-5203422c
{
    "StoppingInstances": [
        {
            "InstanceId": "i-5203422c",
            "CurrentState": {
                "Code": 64,
                "Name": "stopping"
            },
            "PreviousState": {
                "Code": 16,
                "Name": "running"
            }
        }
    ]
}
$ aws ec2 start-instances --instance-ids i-5203422c
$ aws ec2 reboot-instances --instance-ids i-5203422c
$ aws ec2 terminate-instances --instance-ids i-5203422c
```

Spot Quiz (ICON)