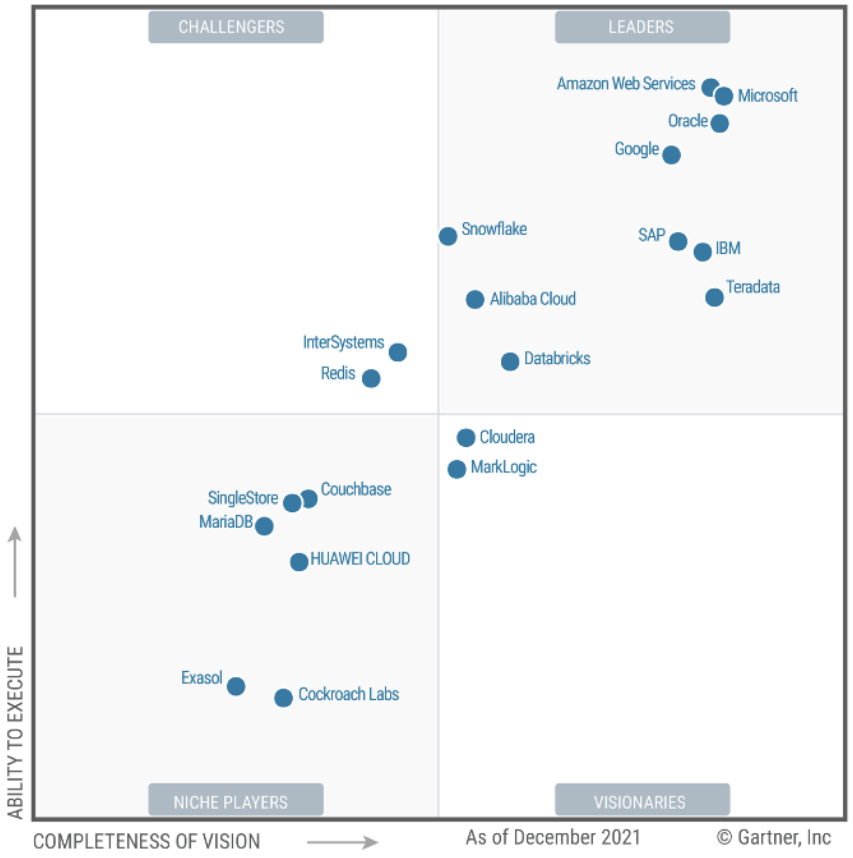
Tutorial 5
Amazon Web Services

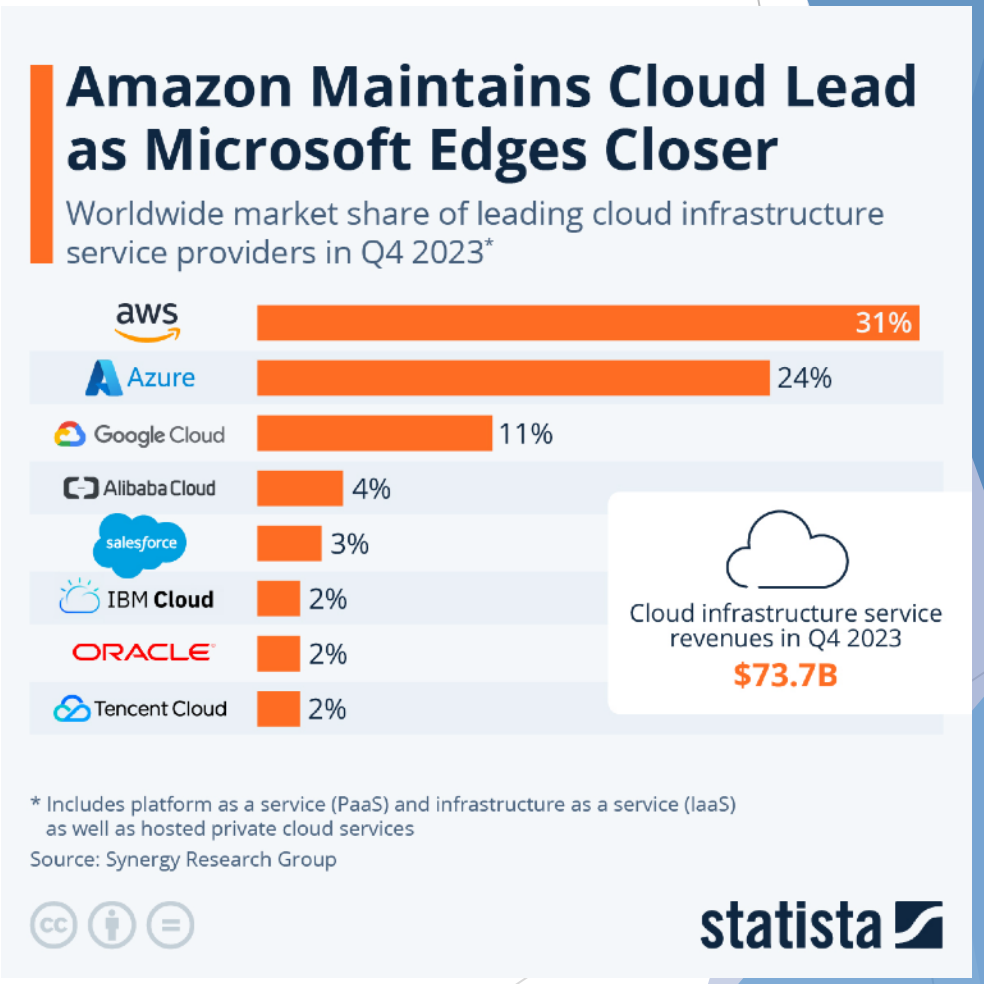
COMP3358 Distributed and Parallel Computing

AWS Overview

Figure 1: Magic Quadrant for Cloud Database Management Systems



Source: Gartner (December 2021)



Sign up for an AWS account

- Open https://aws.amazon.com/
- Choose **Create an AWS Account**, and then follow the online instructions
- You receive the benefits of the free tier automatically for 12 months after you sign up for an AWS account. For more information, see AWS Free Usage Tier

Launch a 64-bit Linux micro-instance

- The AWS management console is a central location where you can create, release, interact with, and monitor your AWS resources. It has a nice GUI, and helpful wizards for performing common tasks (eg, launching EC2 instances, creating SQS queues, creating buckets in S3). Take some time to look around and see what is available
- Helpful resources for this step: http://docs.amazonwebservices.com/AWSEC2/latest/GettingStartedGuide/
- Go to AWS management console (http://aws.amazon.com/console)
- Under EC2 tab, click “Launch Instance”
- Select “Amazon Linux AMI”
- Select to run 1 micro-instance (don’t need to specify availability zone)
- Select defaults on “Advanced Instance Options” page
- Give your new instance a name by specifying a value of the “Name” key
- Create a new secret-key pair, and download the private key to a location on your machine
- Configure your security group: Security groups offer a way to restrict the incoming traffic to a machine, by port, protocol, and IP address. When configuring your security group, ensure you allow access to the machine via the ports below:
  - SSH over port 22
  - FTP over port 21 (custom TCP rule)
  - HTTP over port 80
  - HTTPS over port 443
- Review your details, and click “Launch”. This will launch your first EC2 instance

SSH into your machine

- In order to SSH into your machine, you will need the public DNS name for your EC2 instance. To get the name, go to the “Instances” tab in the AWS management console and select your newly launched instance. The details section should appear at the bottom of the screen, and the public DNS name will be listed there
- For details on how to SSH Into the machine, go here (http://docs.amazonwebservices.com/AWSEC2/latest/GettingStartedGuide/) and navigate to the “Connect to Linux/UNIX instance” page
  - You will need the private key you downloaded above.
  - Log on as “ec2-user”

FTP Into your machine

- Install FTP Server in your AWS instance (https://documentation.ubuntu.com/server/how-to/networking/ftp/index.html)
- Using FTP client, connect to the EC2 machine
  - Specify public DNS name
  - Enter “ec2-user” as user name
  - Specify “Private Key file”
- You can now easily explore your EC2 machine’s file system, copy files to it, move files around, delete files, etc

Another Choice: SCP

- scp -i MyKeyFile.pem FileToUpload.pdf ubuntu@ec2-123-123-123-123.compute-1.amazonaws.com:FileToUpload.pdf

Set-up AWS SDK on your personal machine/AWS Instance

Official Guides:

https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html#setup-install

To successfully develop applications that access AWS services using the AWS SDK for Java, the following conditions are required:

- The Java SDK must have access to credentials to authenticate requests on your behalf.
- The permissions of the IAM role configured for the SDK must allow access to the AWS services that your application requires. The permissions associated with the **PowerUserAccess** AWS managed policy are sufficient for most development needs.
- A development environment with the following elements:
  - Shared configuration files that are set up in at least one of the following ways:
    - The `config` file contains IAM Identity Center single sign-on settings so that the SDK can get AWS credentials.
    - The `credentials` file contains temporary credentials.
  - An installation of Java 8 or later.
  - A build automation tool such as Maven or Gradle.
  - A text editor to work with code.
  - (Optional, but recommended) An IDE (integrated development environment) such as IntelliJ IDEA, Eclipse, or NetBeans.

    When you use an IDE, you can also integrate AWS Toolkits to more easily work with AWS services. The AWS Toolkit for IntelliJ and AWS Toolkit for Eclipse are two toolkits that you can use for Java development.
- An active AWS access portal session when you are ready to run your application. You use the AWS Command Line Interface to initiate the sign-in process to IAM Identity Center's AWS access portal.

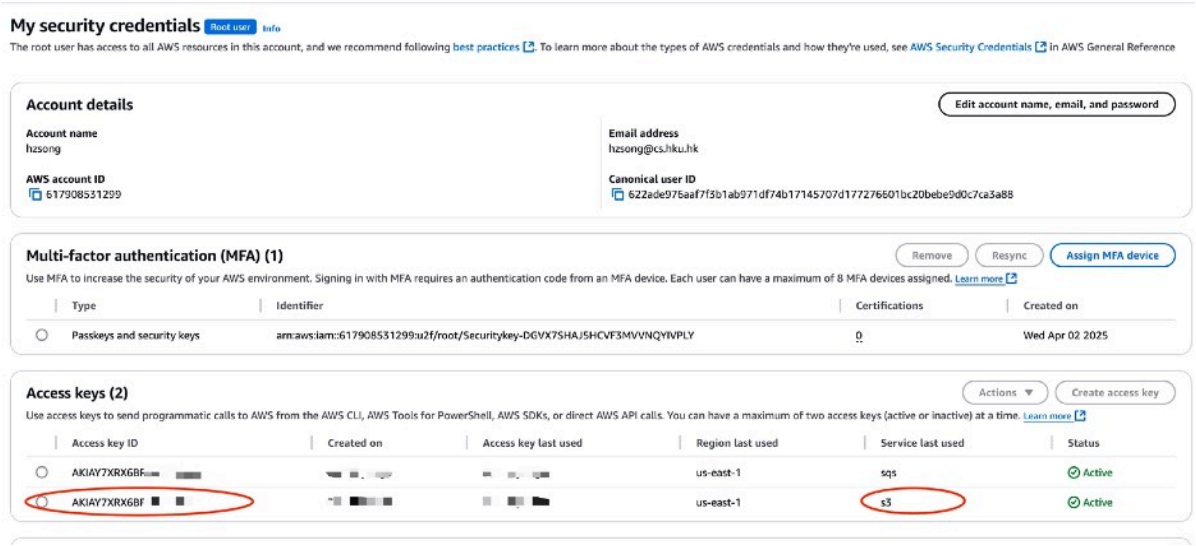
You can use Maven as the build tool.

Set-up AWS SDK on your personal machine

Set up authentication.

```
leo@ubuntu:~$ aws configure
AWS Access Key ID [None]:
```

In AWS Console (IAM) .



"mvn clean package"
“java -jar target/getstarted-1.0-SNAPSHOT.jar”

Exercise

- Write a simple Java application which includes some AWS API calls. You should reference the AWS SDK: https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html#get-started-code
- You can also refer to the demo code on Moodle.
- Submit a doc onto Moodle. The doc should contain the screen shots of your jar execution on AWS. Using the demo code directly or generate a code demo by yourself according to the AWS tutorial is both OK.

AWS S3

- Amazon Simple Storage Service (Amazon S3) is an object storage service.
- https://aws.amazon.com/s3/getting-started/