

## AWS services and working in pairs

In the CAB432 AWS account some services are tied to your username and special steps need to be taken to give your partner access. See [AWS working with a partner on AWS services \(https://canvas.qut.edu.au/courses/20367/pages/working-with-a-partner-on-aws-services\)](https://canvas.qut.edu.au/courses/20367/pages/working-with-a-partner-on-aws-services) for more information.

In this assignment, each criterion has its own list of approved AWS services. You should use only the services listed for that specific criterion, and avoid using others unless you have received prior approval.

**Services for general purpose are approved across the board:** EC2 instances, Route53, S3, EFS, RDS, DynamoDB, CDK, SDK, CloudFormation, Parameter Store, Secrets Manager, Cognito

The page AWS Services Available provides a list of all AWS services available in the CAB432 cloud account.

Services that implement the substantially the same functionality won't count separately as there is little need to have two tools for the same job. For example, if you were to deploy your own instance of MySQL on EC2 *and* use RDS then there is no added value here; it would be better to implement both databases using the same service.

## Core criteria (10 marks)

These criteria relate to core learning outcomes around cloud architecture, scalability, and security. You must attempt every core criterion.

Although there are no criteria related to deployment, it is expected that you will deploy your application to our AWS account.

Note that in this assessment the core criteria represent a smaller portion of the total grade than in the first assessment. This is because we expect that there will be a wide variety of projects, and not all of the cloud services we have studied so far will be appropriate for all projects. Hence the additional criteria, where you have the choice to attempt criteria that are most appropriate to your project, make up a larger portion than in the previous assessment.

### Microservices (3 marks)

For this core criterion we require that your application has at least two separate services running on separate compute.

The separation into two services must be appropriate and not arbitrary. For example, one service might provide the main REST API or serve the web client while the other service implements the CPU intensive process that you will horizontally scale.

The two services must run on separate compute instances, eg. separate EC2 instances, separate containers on ECS, or one running on ECS and the other on EC2.

**Approved services for this criterion:** EC2, ECS

### Load distribution (2 marks)

Your application must use an appropriate mechanism for distributing load to multiple instances of the service implementing your CPU intensive process.

An application load balancer may be suitable if a single instance of the service can handle multiple requests simultaneously.

A message queue accessed by multiple instances, but delivering each message to only one instance (eg. SQS) is often more suitable for processes that require more time or can only handle one task at a time.

**Approved services for this criterion:** Any type of load balancer, SQS

### Auto scaling (3 marks)

The service handling the CPU intensive task in your application must automatically scale horizontally in response to load.

- Clients should not see interruptions in service while scaling out happens, although there may be a graceful degradation in service (eg. response times increase until new instances come online)
- The service can be deployed on ECS or EC2. It is not acceptable to use Lambda for the CPU intensive task service.
- You will need to demonstrate autoscaling from 1 instance/container up to 3 in response to load, and back down to 1 instance when load is reduced.
- The default choice for scaling metric is average CPU utilisation with the target set to 70%. It is not acceptable to reduce the target to compensate for an inability to achieve high average CPU utilisation. If you are using a custom metric then the scaling must be successful with that metric instead.

For EC2, take care that if you are using a single-threaded application that you use a single-CPU instance type (eg. `t2.micro`) and that the *Credit specification* is set to *unlimited* for `t2` instance types.

**Approved services for this criterion:** Auto-scaling groups (EC2), Target groups, Application Auto Scaling (ECS), CloudWatch, Lambda (for custom metric)

### HTTPS (2 marks)

Your application is accessible on the public internet over HTTPS with a valid certificate. This requires:

- You have set up a subdomain of `cab432.com` in Route 53 with a CNAME record pointing to an appropriate endpoint for your application. This is not assessed here as it was assessed in A2, but it is required in order to obtain a certificate.
- You have requested and obtained a certificate using ACM.
- You are using an API Gateway or Application Load Balancer configured to use the certificate and route requests to your server instance(s) serving the publicly accessible web client and REST API.

It is acceptable to use an API Gateway or Application Load Balancer for this purpose even if you are not using other functionality that they provide. But note that in that case they don't count towards additional communication mechanism in the criteria below.

**Approved services for this criterion:** Route53, ALB, API Gateway, CloudFront, Certificate Manager

## Additional criteria (14 marks)

We have provided an excessive number of additional criteria. You do not need to attempt all of them. Keep in mind that we will stop marking once we have considered enough additional criteria to account for 14 marks, regardless of whether you have earned the full 14 marks. There is also an "upon request" that requires approval by the unit coordinator.

You cannot achieve more than 14 marks from these tasks. We will mark only those that you explicitly tell us to consider. You should choose the most appropriate for your application and those you will achieve the best outcome for.

More details are given in the marking rubric. Be sure you are completing the tasks in such a way that the marking rubric is satisfied.

It is not expected that you can respond to all additional criteria as several of them depend on the details of your application.

## Additional microservices (2 marks)

This criterion is about adding more microservices. Attempting this criteria means you will have at least four microservices in total.

The division into multiple services should be appropriate, not arbitrary. Each service should be deployed on its own compute.

**Approved services for this criterion:** Same as the associated core criterion

## Serverless functions (2 marks)

Your application uses Lambda to deploy one or more services. Lambda must be an appropriate choice for the services being deployed in this manner. Some good examples include:

- implementing a custom mechanism for autoscaling
- responding to events, eg. queueing up processing tasks when a client has finished uploading a file to S3 via a pre-signed URL
- lightweight public-facing services

It is not acceptable to use Lambda for your application's CPU intensive task.

**Approved services for this criterion:** Lambda, EventBridge

### Container orchestration with ECS (2 marks)

This criterion is looking for use of ECS to deploy at least one microservice.

**Approved services for this criterion:** ECS

### Advanced container orchestration with ECS (2 marks)

In this additional criteria we will look for use of advanced ECS features. We expect at least two additional functions of: service discovery, rolling updates with failure detection, or tasks launched in response to events or on a schedule.

**Approved services for this criterion:** ECS

### Communication mechanisms (2 marks)

Your application uses additional communication services beyond what is used for load distribution. That could include queues, API gateways, using routing functionality in an application load balancer, publication/subscription mechanisms, etc. The communication mechanisms used should be appropriate to the task.

Using API gateway style functionality in an application load balancer counts separately from its load balancing functionality. i.e., using listener rules to route traffic based on path/method/etc. counts as a separate communication mechanism.

Note that an API gateway or application load balancer used *solely* for implementing TLS does not count towards this criterion.

**Approved services for this criterion:** SQS, API Gateway, load balancers, EventBridge

### Custom scaling metric (2 marks)

Your application must use an *appropriate* scaling metric other than average CPU utilisation to improve autoscaling.

We are looking for:

- Appropriate scaling and load distribution
  - The chosen mechanism should handle increasing traffic without overloading instances or causing service interruptions.

- Improvement over average CPU utilisation
  - The metric should better match your application's needs, leading to improvements such as faster client responses or more efficient cloud resource use.
- Scalability across different sizes
- The mechanism should work effectively whether your application runs on a single instance or scales up to hundreds of instances.

*Note: "Scaling mechanism" is used broadly here. You may use a target metric, simple scaling, or step scaling.*

**Approved services for this criterion:** CloudWatch, Lambda

### Infrastructure as code (2 marks)

For this criterion you should aim to use IaC to deploy the AWS services that your application uses relating to core and additional criteria for this assessment.

We won't evaluate deployment for services covered in Assessment 1 or 2. Basically, this means that services listed under Block 1 or Block 2 on AWS Services Available.

**Approved services for this criterion:** Terraform, CDK, CloudFormation

For other technologies, please ask the teaching team.

### Dead letter queue (2 marks)

For this criterion, you should take advantage of the dead letter queue (DLQ) feature of SQS to appropriately handle messages that cannot be processed successfully. It only makes sense to attempt this criterion if you are already using SQS. The intention is to redirect messages that repeatedly fail processing to the DLQ instead of being lost or endlessly retried. You must implement appropriate handling of the messages that end up in the DLQ.

**Approved services for this criterion:** SQS and associated workers that consume messages on the DLQ

### Edge caching (2 marks)

Your application should make *appropriate* use of edge caching with Cloud Front. This means:

- You should have a convincing reason that the data you are caching will be accessed frequently. This does not have to be true *now* but it should be true in an imagined wide-scale deployment of your application.
- The data that you are caching needs to be infrequently changed. For our purposes this basically means that it should be static. Resources such as static front end HTML/CSS/JS/image files (including those built with React and similar) are good candidates.

**Approved services for this criterion:** CloudFront, ElastiCache


Upon request (2 marks)

This additional criteria exists once in the rubric. You must seek approval from a coordinator.

This criteria gives you the opportunity to gain marks for other functionality or aspects that demonstrate a high level of achievement in the project. Be sure to **first** speak to a coordinator if there is something specific that you'd like to do and get additional credit for.

## Write a report (21 marks)

Your report will cover the following topics:

- A description and diagram of the architecture of your application
- Justification for your architecture choices in the design of your application
- A discussion of further development that would be necessary to make your application scalable and secure for a large-scale deployment
- Implications of your architecture choices on sustainability
- Inclusion of a cost estimate, calculated in the [AWS Pricing Calculator](https://calculator.aws/#/)  (<https://calculator.aws/#/>)

Further details on the report can be found at [3.1 Submission specification \(https://canvas.qut.edu.au/courses/20367/pages/3-dot-1-submission-specification\)](https://canvas.qut.edu.au/courses/20367/pages/3-dot-1-submission-specification)

## Technologies and special permission

We will follow the same guidelines as for Assessments 1 and 2. The following technologies do not require special permission:

- Technologies that you already have permission for from assessment 1 or 2
- All AWS services listed in AWS services available.

## Submission

Your submission will have three parts: