

## **AWS Chapter-01: Compute Services**

## Table of Contents

- **EC2**
- **Lightsail**
- **ECR**
- **ECS**
- **EKS**
- **Lambda**
- **Batch**
- **Elastic Beanstalk**
- **Serverless Application Repository**

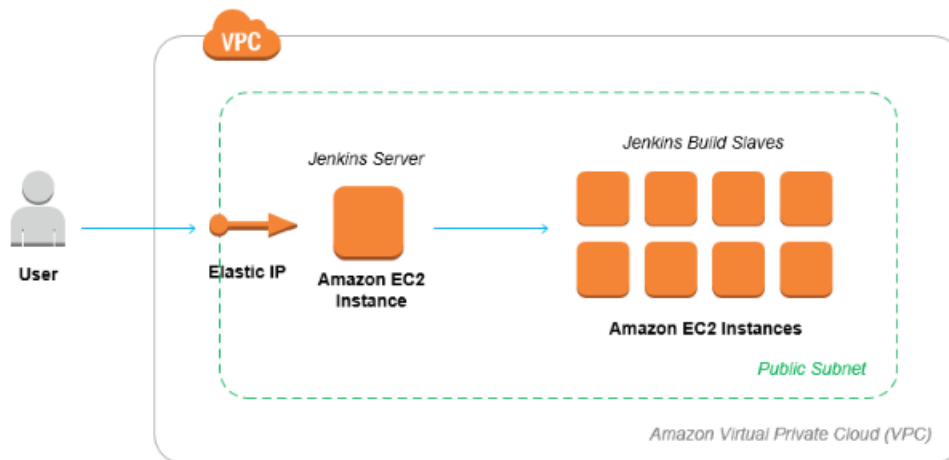
## EC2:

**Amazon** Elastic Compute Cloud (**Amazon EC2**) provides scalable computing capacity in the **Amazon**Web Services (**AWS**) cloud. Using **Amazon EC2** eliminates your need to invest in hardware up front, so you can develop and deploy applications faster.

**For Example :**

### Set Up a Jenkins Build Server

Quickly create a build server for Continuous Integration (CI) on AWS



You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage. Amazon EC2 enables you to scale up or down to handle changes in requirements or spikes in popularity, reducing your need to forecast traffic.

### Features of Amazon EC2

Amazon EC2 provides the following features:

- Virtual computing environments, known as *instances*
- Preconfigured templates for your instances, known as *Amazon Machine Images (AMIs)*, that package the bits you need for your server (including the operating system and additional software)
- Various configurations of CPU, memory, storage, and networking capacity for your instances, known as *instance types*

- Secure login information for your instances using *key pairs* (AWS stores the public key, and you store the private key in a secure place)
- Storage volumes for temporary data that's deleted when you stop or terminate your instance, known as *instance store volumes*
- Persistent storage volumes for your data using Amazon Elastic Block Store (Amazon EBS), known as *Amazon EBS volumes*
- Multiple physical locations for your resources, such as instances and Amazon EBS volumes, known as *Regions* and *Availability Zones*
- A firewall that enables you to specify the protocols, ports, and source IP ranges that can reach your instances using *security groups*
- Static IPv4 addresses for dynamic cloud computing, known as *Elastic IP addresses*
- Metadata, known as *tags*, that you can create and assign to your Amazon EC2 resources
- Virtual networks you can create that are logically isolated from the rest of the AWS cloud, and that you can optionally connect to your own network, known as *virtual private clouds* (VPCs)

## Lightsail:

**Amazon Lightsail** is an **Amazon** cloud service that offers bundles of cloud compute power and memory for new or less experienced cloud users. ... **Amazon Lightsail** launches virtual private servers, which are VMs with individual operating systems but restricted access to physical server resources

Amazon Web Services (AWS) packages memory, processing, storage and transfer into virtual machines (VMs) for customers to purchase, and then releases that compute capacity as Amazon Elastic Compute Cloud (EC2) instances. Amazon Lightsail derives its compute power from an EC2 instance and repackages it for customers who are new or inexperienced with cloud

Plans starting at \$3.50 per month

Lightsail is an easy-to-use cloud platform that offers you everything needed to build an application or website, plus a cost-effective, monthly plan. Whether you're new to the cloud or looking to get on the cloud quickly with AWS infrastructure you trust, we've got you covered.

### No-nonsense monthly pricing

Linux/Unix Windows						
\$3.50 USD	\$5 USD	\$10 USD	\$20 USD	\$40 USD	\$80 USD	\$160 USD
512 MB Memory 1 Core Processor 20 GB SSD Disk 1 TB Transfer*	1 GB Memory 1 Core Processor 40 GB SSD Disk 2 TB Transfer*	2 GB Memory 1 Core Processor 60 GB SSD Disk 3 TB Transfer*	4 GB Memory 2 Core Processor 80 GB SSD Disk 4 TB Transfer*	8 GB Memory 2 Core Processor 160 GB SSD Disk 5 TB Transfer*	16 GB Memory 4 Core Processor 320 GB SSD Disk 6 TB Transfer*	32 GB Memory 8 Core Processor 640 GB SSD Disk 7 TB Transfer*

\* Only outbound data transfer in excess of your plan's data transfer allowance is subject to overage charges. Please see [FAQ](#) for more detail.

#### Data Transfer Allowances: Asia Pacific (Mumbai & Sydney)

Plans in the Mumbai and Sydney Regions include lower data transfer allowances than other regions.

\$3.50	\$5 USD	\$10 USD	\$20 USD	\$40 USD	\$80 USD	\$160 USD
0.5 TB	1 TB	1.5 TB	2 TB	2.5 TB	3 TB	3.5 TB

## Lightsail virtual servers

Lightsail offers virtual servers (instances) that are easy to set up and backed by the power and reliability of AWS. You can launch your website, web application, or project in minutes, and manage your instance from the intuitive Lightsail console or API.

As you're creating your instance, Lightsail lets you click-to-launch a simple operating system (OS) or a pre-configured application or development stack, such as WordPress, Windows, Plesk, LAMP, Nginx, and more.

### Simplified load balancing

Lightsail's simplified load balancing routes web traffic across your instances so that your websites and applications can accommodate variations in traffic, be better protected from outages, and deliver a seamless experience to your visitors.

Additionally, Lightsail load balancers include integrated certificate management, providing free SSL/TLS certificates that can be provisioned and added to a load balancer in just a few clicks. You can request and manage certificates directly from the Lightsail console – and we manage renewals on your behalf.

### Managed databases

Launch a fully configured MySQL or PostgreSQL database in minutes and leave the maintenance to Lightsail.

With Lightsail managed databases, you can easily scale your databases independently of your virtual servers, improve the availability of your applications, or run standalone databases in the cloud. You can also deploy multi-tiered applications, all within Lightsail, by creating multiple instances that are connected to a central managed database, and a load balancer that directs traffic to the instances.

### Upgrade to EC2

As your cloud ideas expand, you can easily move to EC2 with a simple, guided experience. With this feature, Lightsail offers you the comfort of knowing that as you grow your website or application we can help you scale in a way that fits your needs.

Upgrading is simple, you'll take a snapshot of your instance and follow the step-by-step process in the Lightsail console to export your snapshot to EC2. You can then use the Upgrade to EC2 wizard to get your new EC2 instance up and running.

## ECR:

Docker Basics for Amazon **ECR**. Docker is a technology that allows you to build, run, test, and deploy distributed applications that are based on Linux containers. Amazon **ECR** is a managed **AWS** Docker registry service. Customers can use the familiar Docker CLI to push, pull, and manage images.

Docker is available on many different operating systems, including most modern Linux distributions, like Ubuntu, and even Mac OSX and Windows.

### To install Docker on an Amazon EC2 instance

1. Launch an instance with the Amazon Linux 2 AMI. For more information, see *Launching an Instance* in the *Amazon EC2 User Guide for Linux Instances*.
2. Connect to your instance. For more information, see *Connect to Your Linux Instance* in the *Amazon EC2 User Guide for Linux Instances*.
3. Update the installed packages and package cache on your instance.

### **sudo yum update -y**

Install the most recent Docker Community Edition package.

```
sudo amazon-linux-extras install docker
```

Start the Docker service.

```
sudo service docker start
```

Add the ec2-user to the docker group so you can execute Docker commands without using sudo.

```
sudo usermod -a -G docker ec2-user
```

1. Log out and log back in again to pick up the new docker group permissions. You can accomplish this by closing your current SSH terminal window and reconnecting to your instance in a new one. Your new SSH session will have the appropriate docker group permissions.
2. Verify that the ec2-user can run Docker commands without sudo.

```
docker info
```

### **Note**

In some cases, you may need to reboot your instance to provide permissions for the ec2-user to access the Docker daemon. Try rebooting your instance if you see the following error:

```
Cannot connect to the Docker daemon. Is the docker daemon running on this host?
```



## ECS:

Amazon EC2 Container Service (ECS) is a cloud computing service in Amazon Web Services (AWS) that manages containers. It enables developers to deploy and manage scalable applications that run on groups of servers called clusters through application programming interface (API) calls and task definitions. Amazon ECS is a scalable service that is accessible through the AWS Management Console and software developer's kits (SDKs).

Amazon developed ECS in response to the rise of popularity of containerization. ECS enables a developer to specify rules for isolated sets of Elastic Compute Cloud (EC2) instances to increase portability and computing performance by running on top of a host operating system. ECS supports Docker, an open source Linux container service.

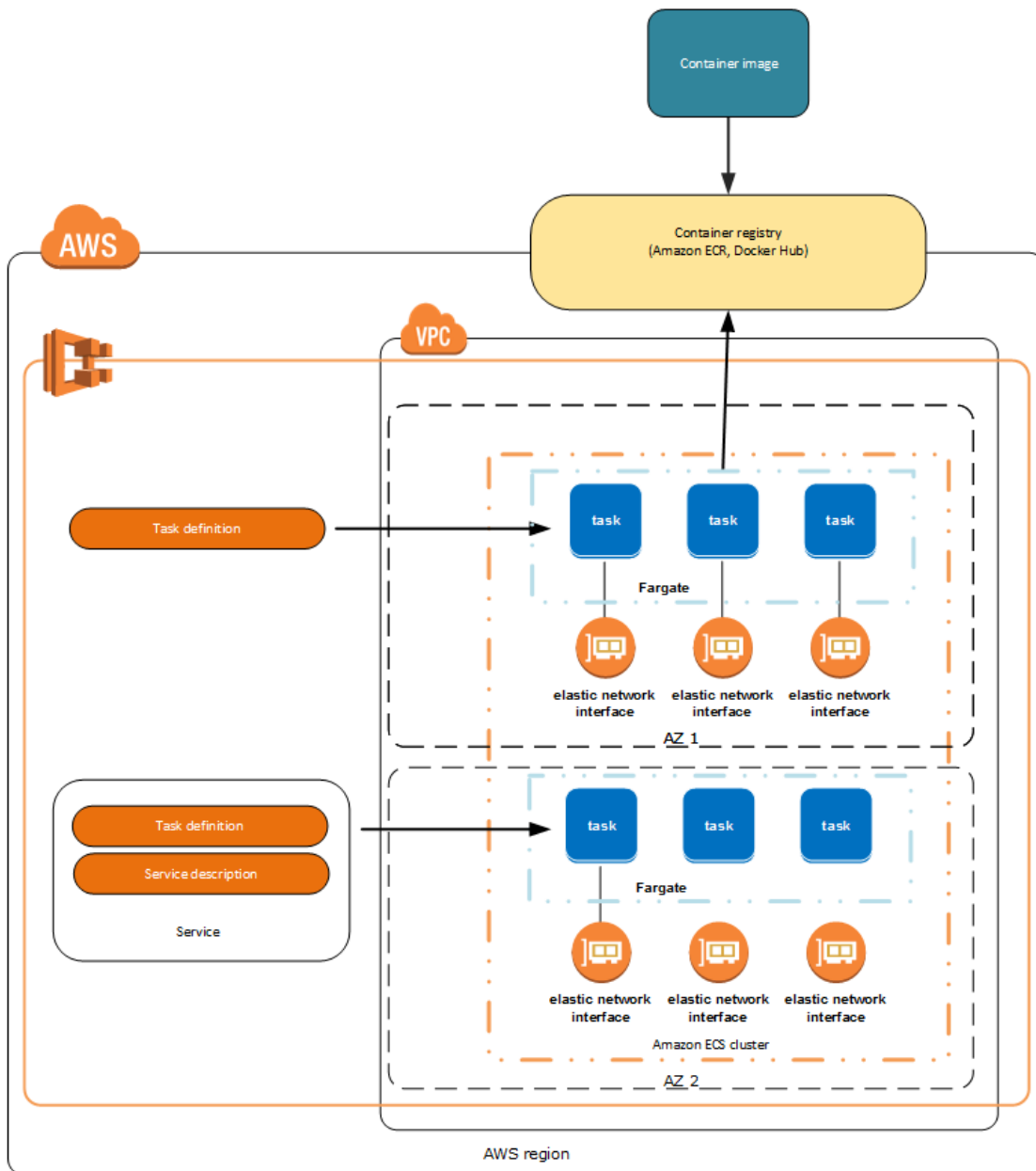
ECS enables users to create and run Docker containers for distributed applications that run on microservices. ECS evaluates and monitors CPU and memory output to determine the optimal deployment for a container. AWS customers can also use the service to update containers or scale them up or down. Elastic Load Balancing, Elastic Block Store (EBS) volumes and Identity and Access Management (IAM) roles are also supported for further customization.

## Amazon EC2 Container Service vs. Kubernetes

Amazon ECS competes with Kubernetes, Google's open source container orchestration system. While the container management tools and use cases differ, Kubernetes has the following features that ECS does not:

- it is deployable to non-AWS clouds and on-premises resources;
- it has storage options outside AWS; and
- it receives contributions from the developer community, while not all ECS code is publicly available.

But Amazon ECS might be a simpler option for businesses that rely on AWS exclusively, or that want a container management platform with easy installation. Load balancer tools, resource monitoring, Auto Scaling and service management features are comparable between the two options.

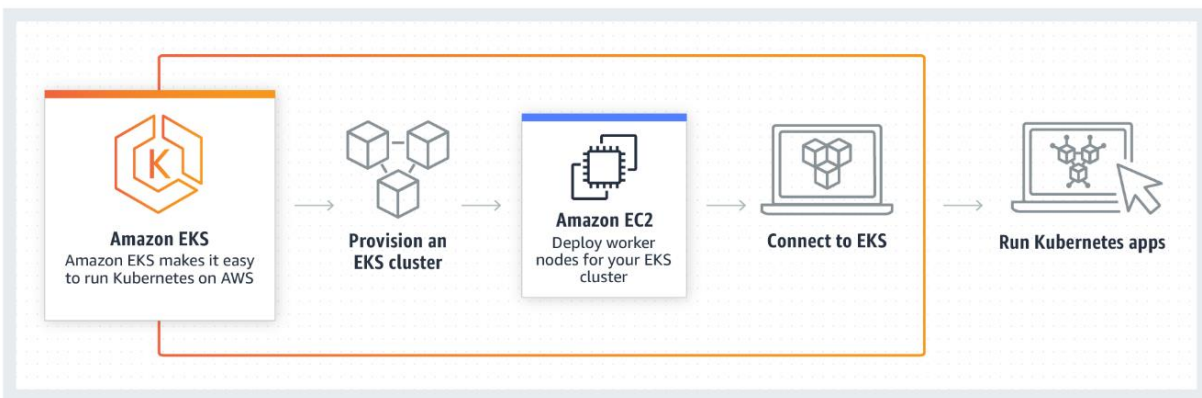


## EKS:

Amazon Elastic Container Service for Kubernetes (Amazon EKS) makes it easy to deploy, manage, and scale containerized applications using [Kubernetes](#) on AWS.

Amazon EKS runs the Kubernetes management infrastructure for you across multiple AWS availability zones to eliminate a single point of failure. Amazon EKS is certified Kubernetes conformant so you can use existing tooling and plugins from partners and the Kubernetes community. Applications running on any standard Kubernetes environment are fully compatible and can be easily migrated to Amazon EKS.

### How it works



### Use Cases

#### MICROSERVICES

Easily run microservices applications with deep integrations to AWS services, while getting access to the full suite of Kubernetes functionality and popular open source tooling.

#### HYBRID CONTAINER DEPLOYMENTS

Run highly available and scalable Kubernetes clusters on AWS while maintaining full compatibility with your Kubernetes deployments running anywhere else.

#### BATCH PROCESSING

The Kubernetes Jobs API lets you run sequential or parallel workloads on your Amazon EKS cluster. These workloads can be run on Amazon EC2 On-Demand Instances, Reserved Instances, or Spot Instances.

## APPLICATION MIGRATION

Easily containerize and migrate existing applications to Amazon EKS without needing to refactor your code or tooling.

### Lambda

AWS Lambda is an event-driven, serverless computing platform provided by Amazon as a part of the Amazon Web Services. It is a computing service that runs code in response to events and automatically manages the computing resources required by that code. It was introduced in November 2014

AWS Lambda is a compute service that lets you run code without provisioning or managing servers. AWS Lambda executes your code only when needed and scales automatically, from a few requests per day to thousands per second. You pay only for the compute time you consume - there is no charge when your code is not running. With AWS Lambda, you can run code for virtually any type of application or backend service - all with zero administration. AWS Lambda runs your code on a high-availability compute infrastructure and performs all of the administration of the compute resources, including server and operating system maintenance, capacity provisioning and automatic scaling, code monitoring and logging

You can use AWS Lambda to run your code in response to events, such as changes to data in an Amazon S3 bucket or an Amazon DynamoDB table; to run your code in response to HTTP requests using Amazon API Gateway; or invoke your code using API calls made using AWS SDKs. With these capabilities, you can use Lambda to easily build data processing triggers for AWS services like Amazon S3 and Amazon DynamoDB, process streaming data stored in Kinesis, or create your own back end that operates at AWS scale, performance, and security.

### When Should I Use AWS Lambda?

AWS Lambda is an ideal compute platform for many application scenarios, provided that you can write your application code in languages supported by AWS Lambda, and run within the AWS Lambda standard runtime environment and resources provided by Lambda.

When using AWS Lambda, you are responsible only for your code. AWS Lambda manages the compute fleet that offers a balance of memory, CPU, network, and other resources. This is in exchange for flexibility, which means you cannot log in to compute instances, or customize the operating system or language runtime. These constraints enable AWS Lambda to perform

operational and administrative activities on your behalf, including provisioning capacity, monitoring fleet health, applying security patches, deploying your code, and monitoring and logging your Lambda functions.

If you need to manage your own compute resources, Amazon Web Services also offers other compute services to meet your needs.

- Amazon Elastic Compute Cloud (Amazon EC2) service offers flexibility and a wide range of EC2 instance types to choose from. It gives you the option to customize operating systems, network and security settings, and the entire software stack, but you are responsible for provisioning capacity, monitoring fleet health and performance, and using Availability Zones for fault tolerance.
- Elastic Beanstalk offers an easy-to-use service for deploying and scaling applications onto Amazon EC2 in which you retain ownership and full control over the underlying EC2 instances.

## **AWS Batch:**

### Introducing AWS Batch

Today I would like to tell you about a new set of fully-managed batch capabilities. [AWS Batch](#) allows batch administrators, developers, and users to have access to the power of the cloud without having to provision, manage, monitor, or maintain clusters. There's nothing to buy and no software to install. AWS Batch takes care of the undifferentiated heavy lifting and allows you to run your container images and applications on a dynamically scaled set of EC2 instances. It is efficient, easy to use, and designed for the cloud, with the ability to run massively parallel jobs that take advantage of the elasticity and selection provided by Amazon EC2 and EC2 Spot and can easily and securely interact with other other AWS services such as Amazon S3, DynamoDB, and SNS.

Let's start by taking a look at some important AWS Batch terms and concepts (if you are already doing batch computing, many of these terms will be familiar to you, and still apply). Here goes:

**Job** – A unit of work (a shell script, a Linux executable, or a container image) that you submit to AWS Batch. It has a name, and runs as a containerized app on EC2 using parameters that you specify in a Job Definition. Jobs can reference other jobs by name or by ID, and can be dependent on the successful completion of other jobs.

**Job Definition** – Specifies how Jobs are to be run. Includes an [AWS Identity and Access Management \(IAM\)](#) role to provide access to AWS resources, and also specifies both memory and CPU requirements. The definition can also control container properties, environment

variables, and mount points. Many of the specifications in a Job Definition can be overridden by specifying new values when submitting individual Jobs.

**Job Queue** – Where Jobs reside until scheduled onto a Compute Environment. A priority value is associated with each queue.

**Scheduler** – Attached to a Job Queue, a Scheduler decides when, where, and how to run Jobs that have been submitted to a Job Queue. The AWS Batch Scheduler is FIFO-based, and is aware of dependencies between jobs. It enforces priorities, and runs jobs from higher-priority queues in preference to lower-priority ones when the queues share a common Compute Environment. The Scheduler also ensures that the jobs are run in a Compute Environment of an appropriate size.

**Compute Environment** – A set of managed or unmanaged compute resources that are used to run jobs. Managed environments allow you to specify desired instance types at several levels of detail. You can set up Compute Environments that use a particular type of instance, a particular model such as c4.2xlarge or m4.10xlarge, or simply specify that you want to use the newest instance types. You can also specify the minimum, desired, and maximum number of vCPUs for the environment, along with a percentage value for bids on the [Spot Market](#) and a target set of VPC subnets. Given these parameters and constraints, AWS Batch will efficiently launch, manage, and terminate EC2 instances as needed. You can also launch your own Compute Environments. In this case you are responsible for setting up and scaling the instances in an [Amazon ECS](#) cluster that AWS Batch will create for you.

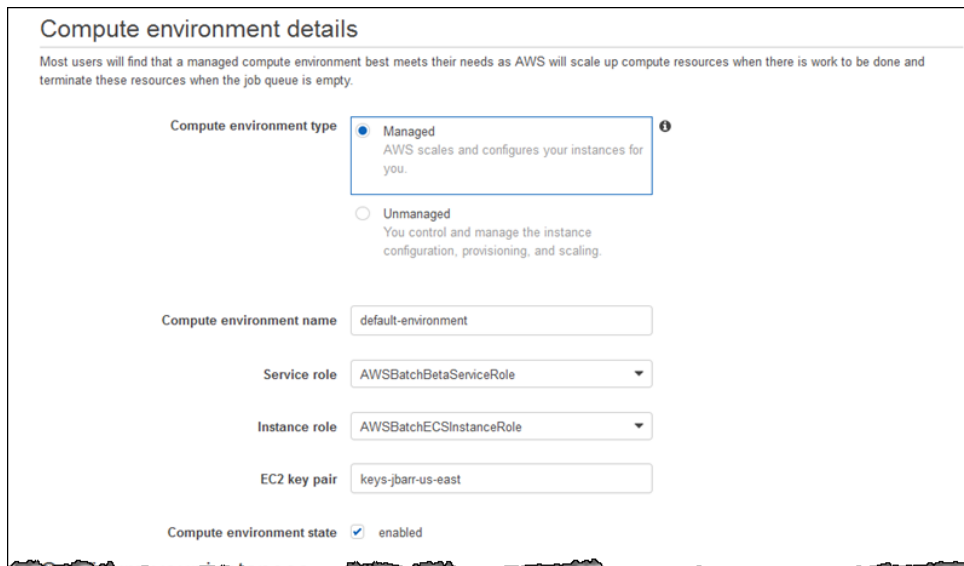
### A Quick Tour

You can access AWS Batch from the [AWS Management Console](#), [AWS Command Line Interface \(CLI\)](#), or via the AWS Batch APIs. Let's take a quick console tour!

The Status Dashboard displays my Jobs, Job Queues, and Compute Environments:

Amazon Newservice		Dashboard					
Dashboard		Job status					
Jobs		<a href="#">Submit job</a>					
Job definitions							
<a href="#">Job queues</a>							
Compute environments							

I need a place to run my Jobs, so I will start by selecting Compute environments and clicking on Create environment. I begin by choosing to create a Managed environment, give it a name, and choosing the IAM roles (these were created automatically for me):



**Compute environment details**

Most users will find that a managed compute environment best meets their needs as AWS will scale up compute resources when there is work to be done and terminate these resources when the job queue is empty.

**Compute environment type**

- ☒ **Managed**  
AWS scales and configures your instances for you.
- ☐ **Unmanaged**  
You control and manage the instance configuration, provisioning, and scaling.

**Compute environment name**

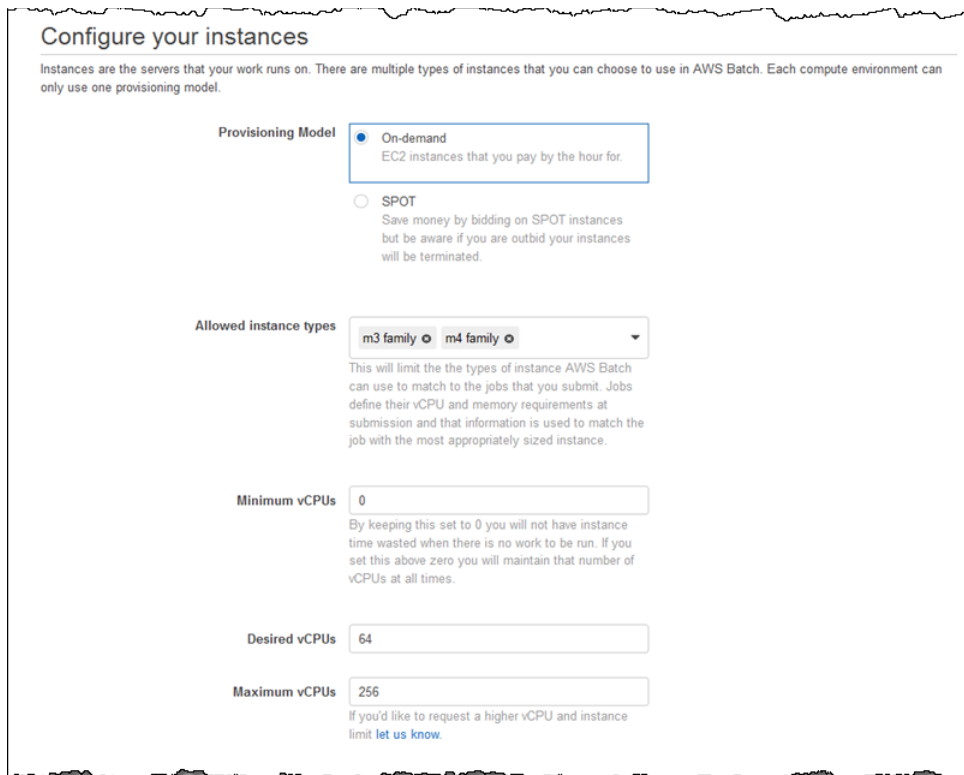
**Service role**

**Instance role**

**EC2 key pair**

**Compute environment state** ☒ enabled

Then I set up the provisioning model (On-Demand or Spot), choose the desired instance families (or specific types), and set the size of my Compute Environment (measured in vCPUs):



**Configure your instances**

Instances are the servers that your work runs on. There are multiple types of instances that you can choose to use in AWS Batch. Each compute environment can only use one provisioning model.

**Provisioning Model**

- ☒ **On-demand**  
EC2 instances that you pay by the hour for.
- ☐ **SPOT**  
Save money by bidding on SPOT instances but be aware if you are outbid your instances will be terminated.

**Allowed instance types**

This will limit the types of instance AWS Batch can use to match to the jobs that you submit. Jobs define their vCPU and memory requirements at submission and that information is used to match the job with the most appropriately sized instance.

**Minimum vCPUs**

By keeping this set to 0 you will not have instance time wasted when there is no work to be run. If you set this above zero you will maintain that number of vCPUs at all times.

**Desired vCPUs**

**Maximum vCPUs**

If you'd like to request a higher vCPU and instance limit [let us know](#).

I wrap up by choosing my VPC, the desired subnets for compute resources, and the security group that will be associated with those resources:

The screenshot shows the 'Networking' section of the 'Create Compute Environment' dialog. The 'VPC Id' is set to 'DefaultVPC | vpc-e68d9c81'. Under 'Subnets', four subnets are selected: 'subnet-75a56749', 'subnet-7216ce5f', 'subnet-009a1149', and 'subnet-b85488e3'. The 'Security groups' dropdown shows 'default | sg-98fa09e5'. The 'Tags' section has a table with one tag: 'Name' with value 'MainCompute'. At the bottom right are 'Cancel' and 'Create' buttons.

Key	Value
Name	MainCompute

I click on Create and my first Compute Environment (MainCompute) is ready within seconds:

The screenshot shows the 'Compute environments' page. It includes a description, buttons for 'Create environment', 'Edit', and 'Delete', a filter input, and a table of existing environments.

Compute environment name	Type	Instance types	Status	State	Min
default-environment	MANAGED	m3,m4	VALID	ENABLED	0

Next, I need a Job Queue to feed work to my Compute Environment. I select Queues and click on Create Queue to set this up. I accept all of the defaults, connect the Job Queue to my new Compute Environment, and click on Create queue:



### Create queue

A queue is a place jobs are submitted to that will then route the job to the appropriate compute environment.

Queue name

Queue status ☒ enabled ⓘ  
When a queue is disabled it will not send jobs to a compute environment but still remains available to receive job submissions.

Priority

#### Connected compute environments for this queue

Work will go to the connected compute environments based on the order they are listed and the available capacity of those environments.

Order	Compute environment
1	Name: default-environment Provisioning model: On-demand

Select a compute environment

[Cancel](#) [Create queue](#)

Again, it is available within seconds:

[Create queue](#) [Edit](#) [Delete](#)

Filter

Queue name	Priority	State	Status	Pending jobs count	Running jobs count
<input checked="" type="radio"/> default-queue	1	ENABLED	VALID	0	0

Now I can set up a Job Definition. I select Job definitions and click on Create, then set up my definition (this is a very simple job; I am sure you can do better). My job runs the `sleep` command, needs 1 vCPU, and fits into 128 MB of memory:

### Create a job definition

Job definitions allow you to save the values for a job so it can be used as a template later.

Job definition name

Job role

Job depends on

Retries

Container image

#### Environment

Command

Parameters

vCPUs

Memory (MB)

uLimits

Limit name	Soft limit	Hard limit
<input type="text" value=""/>	<input type="text" value="10"/>	<input type="text" value="80"/>

[Add limit](#)

I can also pass in environment variables, disable privileged access, specify the user name for the process, and arrange to make file systems available within the container:

#### Environment variables

Key	Value
<input type="text" value="Enter a key"/>	<input type="text" value="Enter a value"/>

[Add Tag](#)

#### Security

Privileged ☐ disabled

User

#### Mount points

Read only filesystem ☐ disabled

Container path	Source path	Read-only
<input type="text" value="Enter a key"/>	<input type="text" value="Enter a value"/>	<input type="checkbox"/> disabled

[Add mount point](#)

[Cancel](#) [Save](#)

I click on Save and my Job Definition is ready to go:

CreateCloneDelete

Filter

Job definition name : revision ▼vCPUs ▼

☐

 SleepJob:11

Now I am ready to run my first Job! I select Jobs and click on Submit job:

Submit an AWS Batch Job

Using this page, you can describe and submit AWS Batch jobs.

Job run-time

Code entry type

Container image ▼

Job name

SleepyJobTake1

Job definition

SleepJob ▼

Job queue

default-queue ▼

Job depends on

optional: enter a jobid this will depend on

Array size

1

Retries

3

I can also override many aspect of the job, add additional tags, and so forth. I'll everything as-is and click on Submit:

### Environment

**Command**

**Parameters**

**vCPUs**   
AWS batch will find a place to run your work that has at least the amount of vCPUs you specify.

**Memory (GB)**   
The amount of memory you need for your job to run.

**uLimits**

Limit name	Soft limit	Hard limit
<input data-bbox="565 653 662 674" type="text" value=""/>	<input data-bbox="678 653 878 674" type="text" value="10"/>	<input data-bbox="906 653 1105 674" type="text" value="80"/>

[Add limit](#)

### Environment variables

Key	Value
<input data-bbox="215 793 699 825" type="text" value="Enter a key"/>	<input data-bbox="751 793 1235 825" type="text" value="Enter a value"/>

[Add Tag](#)

### Tags

Key	Value
<input data-bbox="215 951 699 982" type="text" value="Enter a key"/>	<input data-bbox="751 951 1235 982" type="text" value="Enter a value"/>

[Add Tag](#)

[Cancel](#) [Submit](#)

And there it is:

[Submit job](#)
[Clone job](#)

Queue	Status	Filter			
default-queue	<a href="#">runnable</a> <a href="#">submitted</a> <a href="#">pending</a> <a href="#">starting</a> <a href="#">running</a> <a href="#">succeeded</a> <a href="#">failed</a>	<input type="text" value="Filter"/>	< Viewing 1 - items out of 1 pages >		
Job name	Status	Created at	Array size	Retries	
⌚ SleepyJobTake1	RUNNABLE	10:53:04 am 11/26/16	0	3	

I can also submit jobs by specifying the Ruby, Python, Node, or Bash script that implements the job. For example:

## Submit an AWS Batch Job

Using this page, you can describe and submit AWS Batch jobs.

### Job run-time

Code entry type

Edit inline ▼

Select a language

Ruby ▼

Select a theme

Light ▼

```
1 #!/bin/ruby
2
3 for i in (1..4)
4   print i, " "
5 end
6
```

The command line equivalents to the operations that I used in the console include `create-compute-environment`, `describe-compute-environments`, `create-job-queue`, `describe-job-queues`, `register-job-definition`, `submit-job`, `list-jobs`, and `describe-jobs`.

## Elastic Beanstalk:

AWS Elastic Beanstalk is an orchestration service offered by Amazon Web Services for deploying applications which orchestrates various AWS services, including EC2, S3, Simple Notification Service, CloudWatch, autoscaling, and Elastic Load Balancers.

AWS Elastic Beanstalk is an easy-to-use service for deploying and scaling web applications and services developed with Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker on familiar servers such as Apache, Nginx, Passenger, and IIS.

You can simply upload your code and Elastic Beanstalk automatically handles the deployment, from capacity provisioning, load balancing, auto-scaling to application health monitoring. At the same time, you retain full control over the AWS resources powering your application and can access the underlying resources at any time.

There is no additional charge for Elastic Beanstalk - you pay only for the AWS resources needed to store and run your applications.

### Benefits

#### FAST AND SIMPLE TO BEGIN

Elastic Beanstalk is the fastest and simplest way to deploy your application on AWS. You simply use the AWS Management Console, a Git repository, or an integrated development environment (IDE) such as Eclipse or Visual Studio to upload your application, and Elastic Beanstalk automatically handles the deployment details of capacity provisioning, load balancing, auto-scaling, and application health monitoring. Within minutes, your application will be ready to use without any infrastructure or resource configuration work on your part.

#### DEVELOPER PRODUCTIVITY

Elastic Beanstalk provisions and operates the infrastructure and manages the application stack (platform) for you, so you don't have to spend the time or develop the expertise. It will also keep the underlying platform running your application up-to-date with the latest patches and updates. Instead, you can focus on writing code rather than spending time managing and configuring servers, databases, load balancers, firewalls, and networks.

#### IMPOSSIBLE TO OUTGROW

Elastic Beanstalk automatically scales your application up and down based on your application's specific need using easily adjustable Auto Scaling settings. For example, you can use CPU utilization metrics to trigger Auto Scaling actions. With Elastic Beanstalk, your application can handle peaks in workload or traffic while minimizing your costs.

## COMPLETE RESOURCE CONTROL

You have the freedom to select the AWS resources, such as Amazon EC2 instance type, that are optimal for your application. Additionally, Elastic Beanstalk lets you "open the hood" and retain full control over the AWS resources powering your application. If you decide you want to take over some (or all) of the elements of your infrastructure, you can do so seamlessly by using Elastic Beanstalk's management capabilities.

## Serverless Application Repository:

The AWS Serverless Application Repository is a managed repository for serverless applications. It enables teams, organizations, and individual developers to store and share reusable applications, and easily assemble and deploy serverless architectures in powerful new ways. Using the Serverless Application Repository, you don't need to clone, build, package, or publish source code to AWS before deploying it. Instead, you can use pre-built applications from the Serverless Application Repository in your serverless architectures, helping you and your teams reduce duplicated work, ensure organizational best practices, and get to market faster. Integration with AWS Identity and Access Management (IAM) provides resource-level control of each application, enabling you to publicly share applications with everyone or privately share them with specific AWS accounts.

### How it works: Deploying applications



### BENEFITS

#### Develop More Powerful Apps

Quickly assemble serverless architectures in powerful new ways. Discover and share reusable serverless application patterns, privately or publicly, and compose new serverless architectures using the simplified syntax of AWS SAM.

#### Easily Manage Applications

Use pre-built applications in your serverless deployments, eliminating the need to clone, build, package, and publish source code to AWS before deploying it. It also supports SAM and semantic versioning to enable simple application management.

#### Reuse, Don't Rebuild

Develop and publish serverless applications once, store them in the Serverless Application Repository, and use them privately across teams or with the greater community to reduce duplicated efforts and accelerate development workflows.



### Ensure Best Practices

Find and distribute serverless applications for common use cases. Build organizational best practices into your serverless architectures to help ensure consistency across teams—use permissions to share applications with specific AWS accounts.

Featured Apps: Alexa Skills, IoT, Machine Learning