AWS Batch

Fully managed batch processing at any scale

AWS Batch enables developers, scientists, and engineers to easily and efficiently run hundreds of thousands of batch computing jobs on AWS. AWS Batch dynamically provisions the optimal quantity and type of compute resources (e.g., CPU or memory optimized instances) based on the volume and specific resource requirements of the batch jobs submitted. With AWS Batch, there is no need to install and manage batch computing software or server clusters that you use to run your jobs, allowing you to focus on analyzing results and solving problems. AWS Batch plans, schedules, and executes your batch computing workloads across the full range of AWS compute services and features, such as [AWS Fargate](https://aws.amazon.com/fargate/), [Amazon EC2](https://aws.amazon.com/ec2/) and [Spot Instances](https://aws.amazon.com/ec2/spot/).

There is no additional charge for AWS Batch. You only pay for the AWS resources (e.g. EC2 instances or Fargate jobs) you create to store and run your batch jobs.

# Benefits

### Fully managed

AWS Batch eliminates the need to operate third-party commercial or open source batch processing solutions. There is no batch software or servers to install or manage. AWS Batch manages all the infrastructure for you, avoiding the complexities of provisioning, managing, monitoring, and scaling your batch computing jobs.

### Integrated with AWS

AWS Batch is natively integrated with the AWS platform, allowing you to leverage the scaling, networking, and access management capabilities of AWS. This makes it easy to run jobs that safely and securely retrieve and write data to and from AWS data stores such as Amazon S3 or Amazon DynamoDB. You can also run AWS Batch on AWS Fargate, for fully serverless architecture, eliminating the need to manage compute infrastructure.

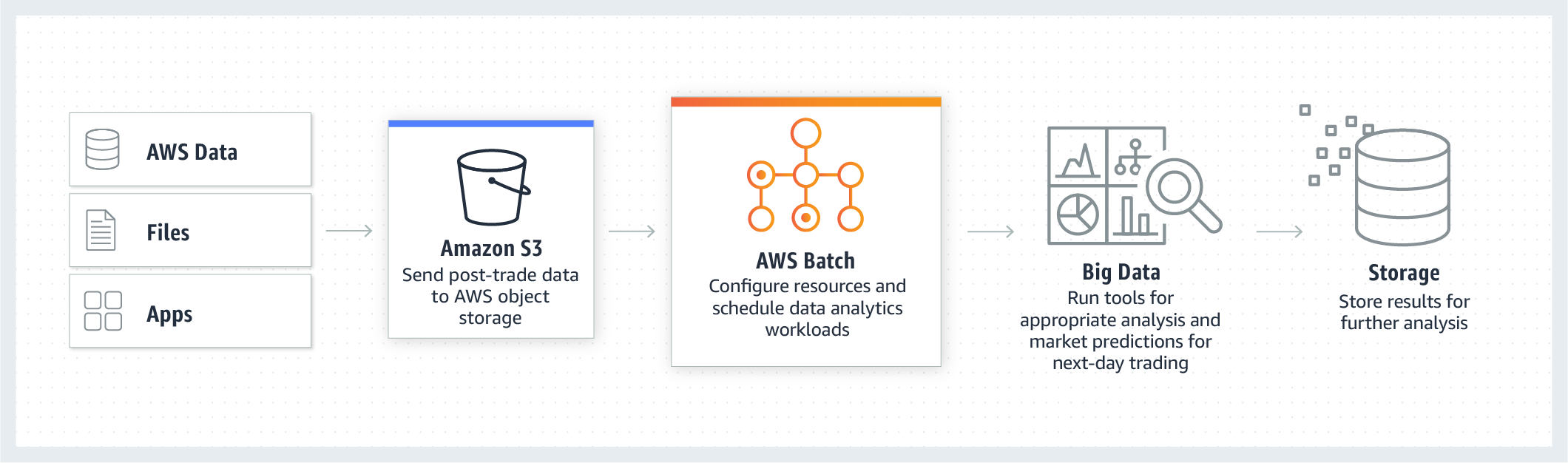
### Cost optimized resource provisioning

AWS Batch provisions compute resources and optimizes the job distribution based on the volume and resource requirements of the submitted batch jobs. AWS Batch dynamically scales compute resources to any quantity required to run your batch jobs, freeing you from the constraints of fixed-capacity clusters. AWS Batch will utilize Spot Instances or submit to Fargate Spot on your behalf, reducing the cost of running your batch jobs further.

## **Use cases**

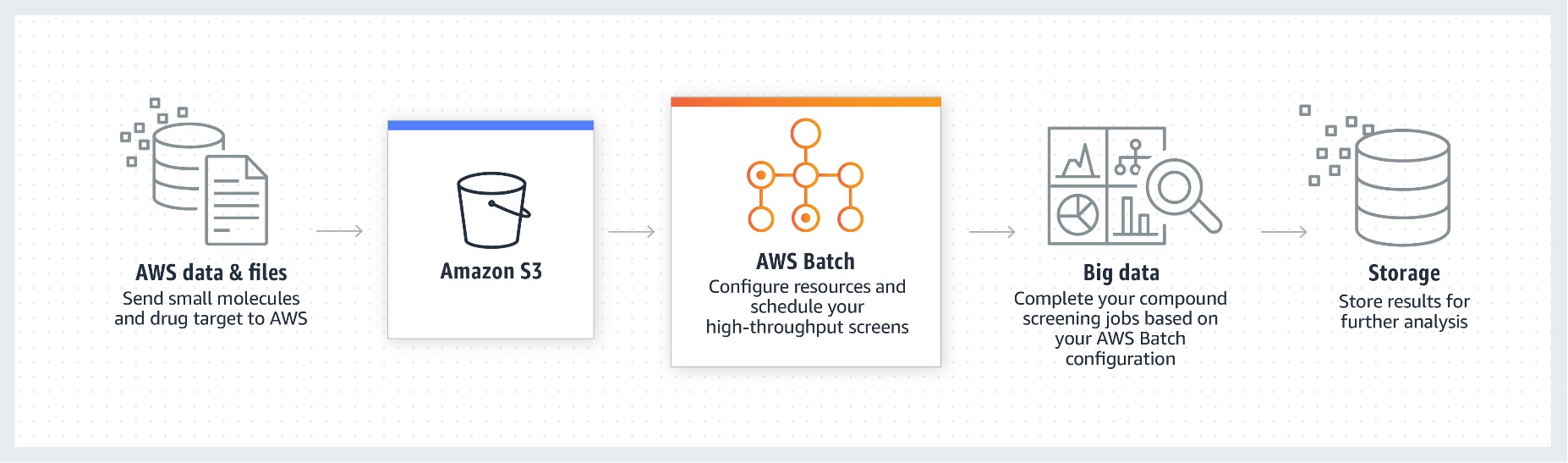
### Financial services: Post-trade analytics

Automate the analysis of the day’s transaction costs, execution reporting, and market performance.



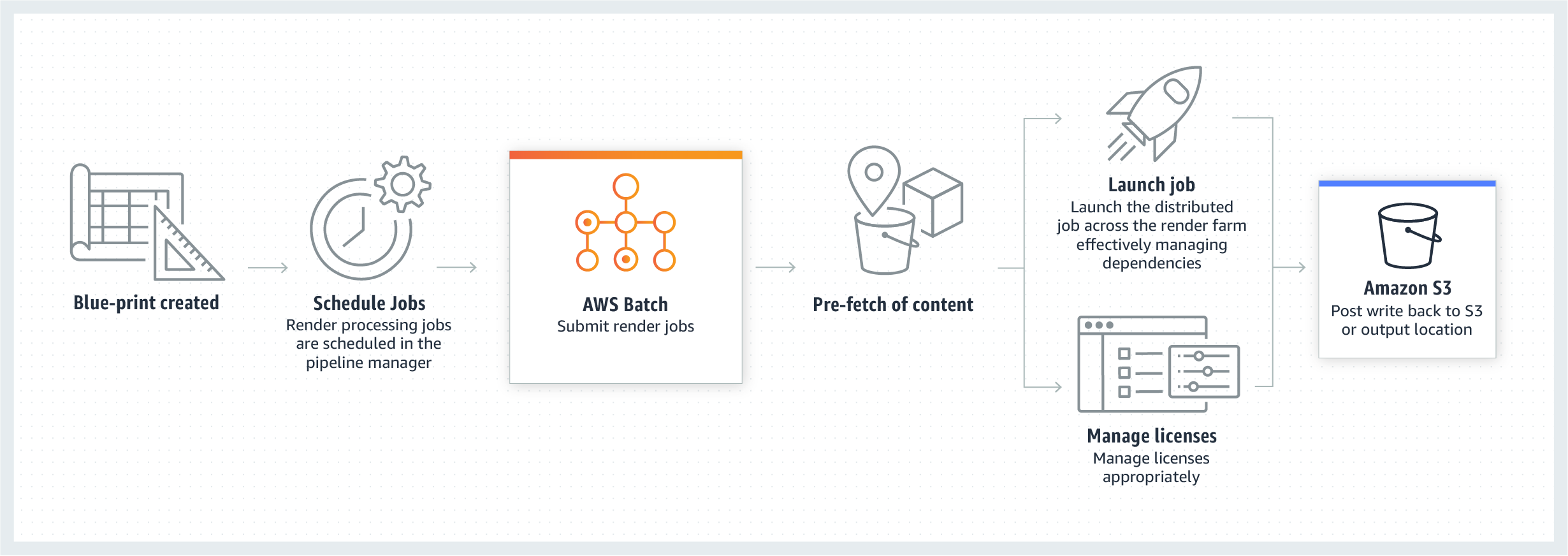
### Life sciences: Drug screening for biopharma

Rapidly search libraries of small molecules for drug discovery.



### Digital media: Visual effects rendering

Automate content rendering workloads and reduce the need for human intervention due to execution dependencies or resource scheduling.



# AWS Batch features

# With AWS Batch, you simply package the code for your batch jobs, specify their dependencies, and submit your batch job using the AWS Management Console, CLIs, or SDKs. AWS Batch allows you to specify execution parameters and job dependencies, and facilitates integration with a broad range of popular batch computing workflow engines and languages (e.g., Pegasus WMS, Luigi, Nextflow, Metaflow, Apache Airflow, and AWS Step Functions). AWS Batch efficiently and dynamically provisions and scales [Amazon EC2](https://aws.amazon.com/ec2/) and [Spot](https://aws.amazon.com/ec2/spot/) Instances or leverages [Fargate](https://aws.amazon.com/fargate/) and Fargate Spot based on the requirements of your jobs. AWS Batch provides default job queues and compute environment definitions that enable you to get started quickly.

## **Dynamic compute resource provisioning and scaling**

When using Fargate or Fargate Spot with Batch, you only need to set up a few concepts in Batch (a CE, job queue, and job definition), and you have a complete queue, scheduler, and compute architecture without managing a single piece of compute infrastructure.

For those wanting EC2 instances, AWS Batch provides Managed Compute Environments that dynamically provision and scale compute resources based the volume and resource requirements of your submitted jobs. You can configure your AWS Batch Managed Compute Environments with requirements such as type of EC2 instances, VPC subnet configurations, the min/max/desired vCPUs across all instances, and the amount you are willing to pay for Spot Instances as a % of the On-Demand Instance price.

Alternatively, you can provision and manage your own compute resources within AWS Batch Unmanaged Compute Environments if you need to use different configurations (e.g., larger EBS volumes or a different operating system) for your EC2 instances than what is provided by AWS Batch Managed Compute Environments. You just need to provision EC2 instances that include the Amazon ECS agent and run supported versions of Linux and Docker. AWS Batch will then run batch jobs on the EC2 instances that you provision.

## **AWS Batch with Fargate**

AWS Batch with Fargate resources allows you to have a completely serverless architecture for your batch jobs. With Fargate, every job receives the exact amount of CPU and memory that it requests (within allowed Fargate SKU’s), so there is no wasted resource time or need to wait for EC2 instance launches.

If you’re a current Batch user, Fargate allows for an additional layer of separation away from EC2. There’s no need to manage or patch AMI’s. When submitting your Fargate-compatible jobs to Batch, you don’t have to worry about maintaining two different services if you have workloads that run on EC2, and others that run on Fargate.

AWS Provides a cloud-native scheduler complete with a managed queue and the ability to specify priority, retries, dependencies, timeouts, and more. Batch manages submission to Fargate and the lifecycle of your jobs so you don’t have to.

Fargate also provides security benefits that come with no added effort (e.g., SOX, PCI compliance), and isolation between compute resources for every job.

## **Support for tightly-coupled HPC workloads**

AWS Batch supports multi-node parallel jobs, which enables you to run single jobs that span multiple EC2 instances. This feature lets you use AWS Batch to easily and efficiently run workloads such as large-scale, tightly-coupled High Performance Computing (HPC) applications or distributed GPU model training. AWS Batch also supports [Elastic Fabric Adapter](https://aws.amazon.com/hpc/efa/), a network interface that enables you to run applications that require high levels of inter-node communication at scale on AWS.

## **Granular job definitions and simple job dependency modeling**

AWS Batch allows you to specify resource requirements, such as vCPU and memory, [AWS Identity and Access Management](https://aws.amazon.com/iam/) (IAM) roles, volume mount points, container properties, and environment variables, to define how jobs are to be run. AWS Batch executes your jobs as containerized applications running on [Amazon ECS](https://aws.amazon.com/ecs/). Batch also enables you to define dependencies between different jobs. For example, your batch job can be composed of three different stages of processing with differing resource needs. With dependencies, you can create three jobs with different resource requirements where each successive job depends on the previous job.

## **Priority-based job scheduling**

AWS Batch enables you to set up multiple queues with different priority levels. Batch jobs are stored in the queues until compute resources are available to execute the job. The AWS Batch scheduler evaluates when, where, and how to run jobs that have been submitted to a queue based on the resource requirements of each job. The scheduler evaluates the priority of each queue and runs jobs in priority order on optimal compute resources (e.g., memory vs CPU optimized), as long as those jobs have no outstanding dependencies.

## **Support for GPU scheduling**

GPU scheduling allows you to specify the number and type of accelerators your jobs require as job definition input variables in AWS Batch. AWS Batch will scale up instances appropriate for your jobs based on the required number of GPUs and isolate the accelerators according to each job’s needs, so only the appropriate containers can access them.

## **Support for popular workflow engines**

AWS Batch can be integrated with commercial and open-source workflow engines and languages such as Pegasus WMS, Luigi, Nextflow, Metaflow, Apache Airflow, and AWS Step Functions, enabling you to use familiar workflow languages to model your batch computing pipelines.

## **Integration with EC2 Launch Templates**

AWS Batch now supports EC2 Launch Templates, allowing you to build customized templates for your compute resources, and enabling Batch to scale instances with those requirements. You can specify your EC2 Launch Template to add storage volumes, specify network interfaces, or configure permissions, among other capabilities. EC2 Launch Templates reduce the number of steps required to configure Batch environments by capturing launch parameters within one resource.

## **Flexible allocation strategies**

AWS Batch allows customers to choose three methods to allocate compute resources. These strategies allow customers to factor in throughput as well as price when deciding how AWS Batch should scale instances on their behalf.

Best Fit: AWS Batch selects an instance type that best fits the needs of the jobs with a preference for the lowest-cost instance type. If additional instances of the selected instance type are not available, AWS Batch will wait for the additional instances to be available. If there are not enough instances available, or if the user is hitting Amazon EC2 service limits then additional jobs will not be run until currently running jobs have completed. This allocation strategy keeps costs lower but can limit scaling.

Best Fit Progressive: AWS Batch will select additional instance types that are large enough to meet the requirements of the jobs in the queue, with a preference for instance types with a lower cost per unit vCPU. If additional instances of the previously selected instance types are not available, AWS Batch will select new instance types.

Spot Capacity Optimized: AWS Batch will select one or more instance types that are large enough to meet the requirements of the jobs in the queue, with a preference for instance types that are less likely to be interrupted. This allocation strategy is only available for Spot Instance compute resources.

## **Integrated monitoring and logging**

AWS Batch displays key operational metrics for your batch jobs in the AWS Management Console. You can view metrics related to compute capacity, as well as running, pending, and completed jobs. Logs for your jobs (e.g., STDERR and STDOUT) are available in the AWS Management Console and are also written to Amazon CloudWatch Logs.

## **Fine-grained access control**

AWS Batch uses IAM to control and monitor the AWS resources that your jobs can access, such as Amazon DynamoDB tables. Through IAM, you can also define policies for different users in your organization. For example, admins can be granted full access permissions to any AWS Batch API operation, developers can have limited permissions related to configuring compute environments and registering jobs, and end users can be restricted to the permissions needed to submit and delete jobs.

# Useful links: [Getting Started Guide](http://docs.aws.amazon.com/batch/latest/userguide/Batch_GetStarted.html), [AWS Batch FAQs](https://aws.amazon.com/batch/faqs/?nc1=h_ls)

# 