



Web



Deployment



WEB > GUIDES > PSPDFKIT SERVER > DEPLOYMENT

AWS deployment for PSPDFKit Server



PSPDFKit Server has been deprecated and replaced by [Document Engine](#). To migrate to Document Engine and unlock advanced document processing capabilities, refer to our [migration guide](#). Learn more about these enhancements on our [blog](#).

This guide is designed to provide an overview of how to deploy PSPDFKit Server to AWS using S3, RDS, and ECS. It also includes information on how to set up all the services you need, like:

- ✧ AWS S3 Bucket — Stores all your documents and assets
- ✧ AWS Relational Database Service — Provides the PostgreSQL database needed by PSPDFKit Server
- ✧ AWS Elastic Container Service — Manages container configuration and scaling

Following this guide will give you a completely cloud-based setup of PSPDFKit Server, including a hosted database and all your documents being stored on an S3 bucket.

Note that this guide only represents an example configuration and is by no means an optimal and production-ready way to set up PSPDFKit Server on AWS. As such, you'll probably have to adjust networking and security options according to your requirements.

You'll start by setting up the storage backend and database. You'll continue by setting up an ECS cluster including services, task definitions, and scaling tactics.

Setting up your S3 bucket



ASK AI

Setting up your S3 bucket as your storage backend consists of two steps: creating a bucket, and adding a policy to control which users and roles should have access to the bucket. Create a bucket in the AWS S3 Console, give it a name, and set a region. You can leave all other settings at their default configurations.



You'll need both the region and the name of the bucket later on when configuring your Docker container. Since you don't want your bucket to be publicly accessible, you need to use a policy to grant access to a specific IAM user or role. You can create a minimal user with programmatic access and without any permissions via the IAM console.

Review

Review your choices. After you create the user, you can view and download the autogenerated password and access key.



This user has no permissions

You haven't given this user any permissions. This means that the user has no access to any AWS service or resource. Consider returning to the previous step and adding some type of permissions.

User details

User name	pspdfkit-server
AWS access type	Programmatic access - with an access key

[Cancel](#)[Previous](#)[Create user](#)

After creating the user, you can find the user ARN in the user summary. It'll look something like this:

```
arn:aws:iam::123345789012:user/pspdfkit .
```

You can now select your previously created S3 bucket and go to the permissions tab, where you'll be able to set the bucket policy. You can allow a user full access to the bucket via the following policy, or you can generate your own policy via the [AWS Policy Generator](#):

```
1  {
2    "Version": "2012-10-17",
3    "Statement": [
4      {
5        "Effect": "Allow",
6        "Principal": {
7          "AWS": "<IAM-User-ARN>"
8        },
9        "Action": [
10         "s3:PutObject",
11         "s3:GetObjectAcl",
12         "s3:GetObject",
13         "s3:ListBucket",
14         "s3:DeleteObject"
15       ],
16       "Resource": [ "<S3-Bucket-ARN>", "<S3-Bucket-ARN>/*" ]
17     }
18   ]
19 }
```

After saving the policy, your bucket is ready to be used as your asset storage backend.

Setting up your database

Setting up a PostgreSQL database via AWS RDS is a bit more complicated; not only will you need to create the database, but this database will exist within a VPC and security group, requiring you to set up an inbound rule to allow PSPDFKit Server instances to access your database.

Launch a new PostgreSQL database via the [AWS RDS Console](#) and set a username and password. In the advanced setting, create a new VPC and set a database name. You'll need the database name, username, and password later when configuring PSPDFKit Server. Once the database is launched, you can see its status in the [instances](#) list of the AWS RDS Console.

Click on the newly created database, and then find and make note of the VPC group, as you'll need it when setting up the container.

Setting up the container cluster

The next steps are setting up the cluster that groups and manages container instances and defining tasks that will run PSPDFKit Server. These containers will run images served by our container repository, connect to your PostgreSQL database, and save documents to your S3 bucket.

The first step is creating a cluster on the [AWS ECS Cluster Console](#). Use the EC2 Linux + Networking cluster template, leave the instance configuration as is or adjust it to your liking, and set the VPC to match the one of your database, but create a new security group.

Now you need to create a task definition on the [AWS ECS Task Definition Console](#). Select the EC2 launch type compatibility, give it a name, and select `bridge` as the network mode. You'll also need to assign the task execution role and add a container. To configure the container, click Add Container and set the following options:

- ⚙ Set the image to `public.ecr.aws/pspdfkit/pspdfkit:2024.1.2`.
- ⚙ Map host port `80` to container port `5000`.
- ⚙ Enter the environment variables as explained in the [configuration overview](#) guide and as shown below.

Note that, since this example uses S3 as the storage backend, it's also necessary to set S3-specific configuration options, as described in our [asset storage](#) guide.

Add container

Container name* PSPDFKit-Server 

Image* public.ecr.aws/pspdfkit/pspdfkit:2021.3.0 

Private repository authentication* ☐ 

Memory Limits (MiB)* Soft limit ▼ 512 

[+ Add Hard limit](#)

Define hard and/or soft memory limits in MiB for your container. Hard and soft limits correspond to the `memory` and `memoryReservation` parameters, respectively, in task definitions.

ECS recommends 300-500 MiB as a starting point for web applications.

Port mappings Host port Container port Protocol 

80 5000 tcp 

[+ Add port mapping](#)

If you intend to use an Application Load Balancer (ALB) with your tasks, you can set the host port to 0 to enable dynamic host port mapping. This allows you to run more than one copy of a task on a container instance. [Learn more](#)

ENVIRONMENT

CPU units GPUs Essential ☒Entry point *comma delimited: sh, -c*Command *comma delimited: echo,hello world*Working directory */usr/app*

Environment variables

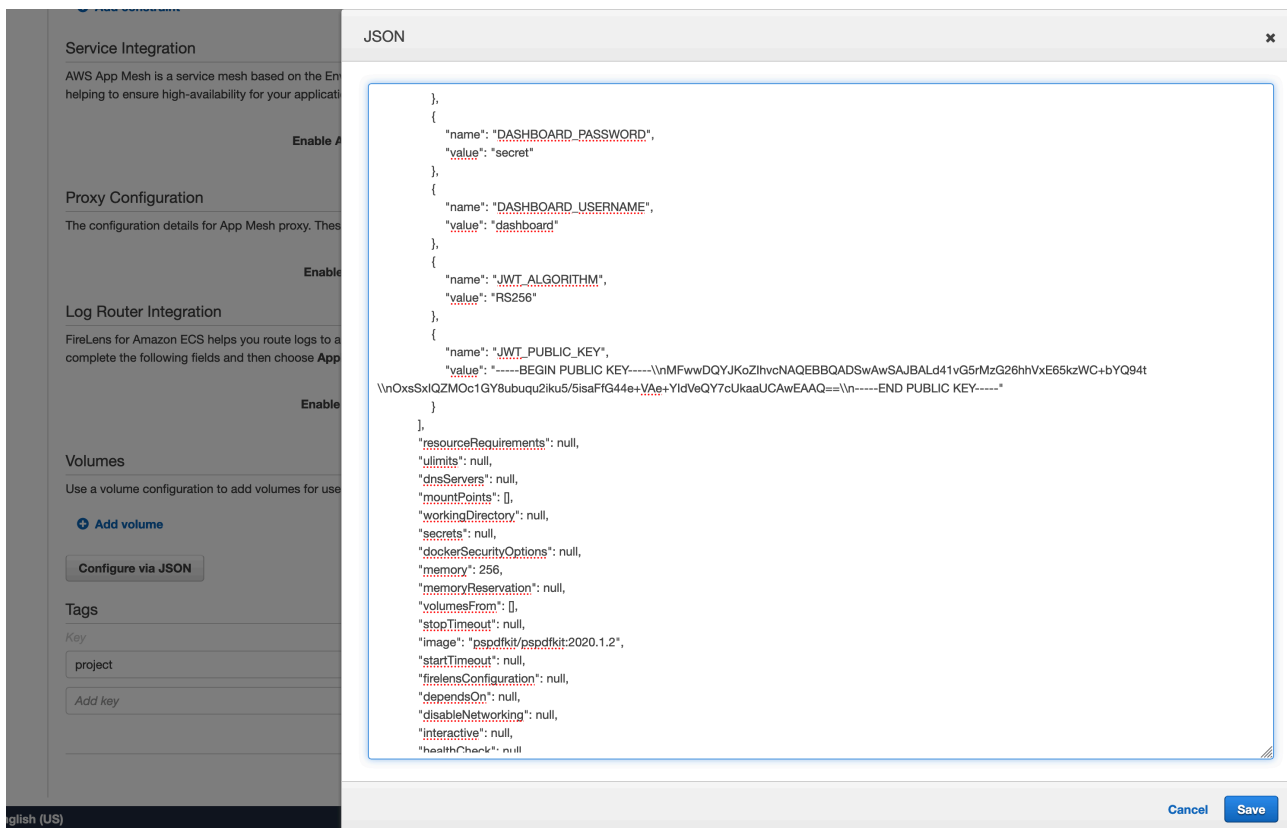
You may also designate AWS Systems Manager Parameter Store keys or ARNs using the 'valueFrom' field. ECS will inject the value into containers at run

Key

ACTIVATION_KEY	Value	ACTIVATION_KEY	✕
API_AUTH_TOKEN	Value	API_AUTH_TOKEN	✕
ASSET_STORAGE_BACKEND	Value	postgres	✕
DASHBOARD_USERNAME	Value	dashboard	✕
DASHBOARD_PASSWORD	Value	pspdfkit	✕
JWT_ALGORITHM	Value	RS256	✕
JWT_PUBLIC_KEY	Value	-----BEGIN PUBLIC KEY-----MI	✕
PGDATABASE	Value	postgres	✕



To correctly paste the public key, it's necessary to use the Configure via JSON option. Additionally, the newline characters of the public key need to be replaced with the `\n` escape sequence.



We strongly recommend enabling logging integration with [AWS CloudWatch](#) so that the server logs are retained across restarts and are easier to search through.

Log configuration ☒ Auto-configure CloudWatch Logs

Log driver

Log options

Key	Value
awslogs-group	/ecs/pspdfkit-server
awslogs-region	eu-central-1
awslogs-stream-pre	ecs
Add key	Add value

The last step is to select your cluster on the [AWS ECS Cluster Console](#) and create a service that actually runs the task. Select the task definition you just created and define the number of tasks you want to run. Select no load balancer and do not activate auto scaling.

If you select your cluster in the [AWS ECS Cluster Console](#), you'll now see a running task in the Tasks tab. Click on it and check out the Logs tab to see your server starting up. If everything is set up correctly, it'll show a message similar to the following: `info] Running server version 2024.1.2, licensed for production use. pid=<0.1912.0>`.

Back in the Details tab, click on the container instance. You can now copy the public DNS value into your browser's address bar to check if your server shows you a `PSPDFKit Server up and running.` message. Try to visit `/dashboard` and log in using the dashboard username and password set in the task definition.

Network settings

Next you'll need to create a new inbound rule for our PostgreSQL database to allow our container instances network access. First you need to find out what security group our EC2 instance is part of, and then you need to allow access to the database from this security group. You can find the security group of your EC2 instance by going to your cluster's EC2 Instances tab in the AWS ECS Cluster Console and clicking on the EC2 instance ID. Take note of the group ID (similar to `sg-123abc4e`) of the security group.

Select your instance in the [AWS RDS Instance Console](#) and look for the Security Group property. Select the security group and add a new inbound rule: Select PostgreSQL as the type, paste the group ID you just noted, and save the newly created rule.

Results

If everything worked out, you're now running a PSPDFKit Server setup using AWS Container Services connected to an AWS-hosted PostgreSQL database and using an S3 bucket as document and asset storage.

You can now start using PSPDFKit for Web, Android, or iOS to integrate document serving and syncing capabilities into your applications.

Limitations

Note that this setup is by no means a one-size-fits-all solution, but rather a minimal setup for using the infrastructure and services provided by AWS to deploy and run PSPDFKit Server. We recommend looking deeper into [AWS](#) to deploy a production-ready setup.

Performance

The performance of a PSPDFKit Server instance on AWS largely depends on the amount of compute resources provided by the underlying compute platform: [AWS EC2](#) or [AWS Fargate](#).

EC2-based services

When using [AWS EC2](#) as an underlying compute platform for ECS, available resources are determined by the instance type you choose. PSPDFKit Server supports both AMD- and Intel-based EC2 instance

types, as well as AWS Graviton-based instances. In general, we suggest choosing Graviton-based instances, as they offer a superior price-to-performance ratio.

Be cautious about using [burstable instances](#) (i.e. T2, T3, and T3a tier). These instance types provide increased performance when the CPU credits are available. However, once credits run out, PSPDFKit Server may not be able to process documents as expected. For example, you may observe that Server errors out when you upload large documents. In such a case, check the Server logs and see if errors overlap with high CPU usage and depleting CPU credits.

To further limit the resources assigned to PSPDFKit Server containers, you can optionally configure the task size in your ECS task definition.

Task size

The task size allows you to specify a fixed size for your task. Task size is required for tasks using the Fargate launch type and is optional for the EC2 launch type. Container level memory settings are optional when task size is set. Task size is not supported for Windows containers.

Task memory (MiB)

4096

The amount of memory (in MiB) used by the task. It can be expressed as an integer using MiB, for example 1024, or as a string using GB, for example '1GB' or '1 gb'.

Task CPU (unit)

2 vCPU

The number of CPU units used by the task. It can be expressed as an integer using CPU units, for example 1024, or as a string using vCPUs, for example '1 vCPU' or '1 vcpu'.

Note that you can't assign more resources than are provided by a single EC2 instance you choose.

Fargate-based services

[AWS Fargate](#) dynamically provisions compute and memory resources you request so that you don't need to manage virtual machines, as is the case in case of EC2. With Fargate, it's necessary that you specify the task size in your ECS task definition.

Task size

The task size allows you to specify a fixed size for your task. Task size is required for tasks using the Fargate launch type and is optional for the EC2 launch type. Container level memory settings are optional when task size is set. Task size is not supported for Windows containers.

Task memory (MiB)

4096

The amount of memory (in MiB) used by the task. It can be expressed as an integer using MiB, for example 1024, or as a string using GB, for example '1GB' or '1 gb'.

Task CPU (unit)

2 vCPU

The number of CPU units used by the task. It can be expressed as an integer using CPU units, for example 1024, or as a string using vCPUs, for example '1 vCPU' or '1 vcpu'.

This amount of resources will be available to the running PSPDFKit Server container.

Container memory limits

Regardless of the launch type used, you can also configure the memory limit for the container itself.

Memory Limits (MiB)*

Hard limit

4096

[+ Add Soft limit](#)

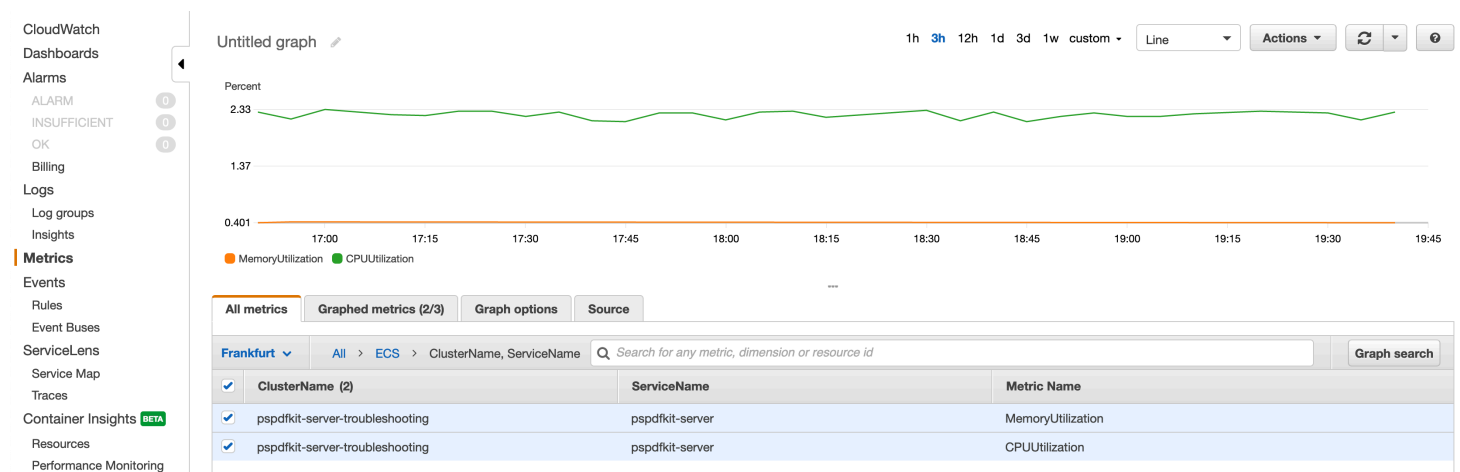
Define hard and/or soft memory limits in MiB for your container. Hard and soft limits correspond to the `memory` and `memoryReservation` parameters, respectively, in task definitions.

ECS recommends 300-500 MiB as a starting point for web applications.

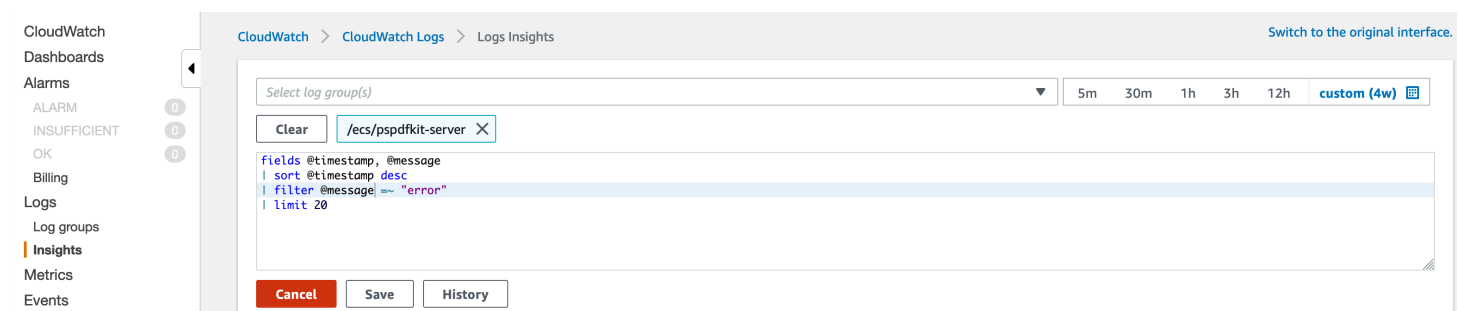
We recommend not setting this limit and instead relying on the task size.

Monitoring with CloudWatch

ECS services on AWS are automatically monitored with [AWS CloudWatch](#). You can use CloudWatch to observe the CPU and memory utilization of your ECS clusters and services.



You might see that high resource utilization aligns with an increased error log rate (if you enabled CloudWatch Logs integration in your container settings).



In this case, it most likely means there are too few resources provisioned for the PSPDFKit Server container for the amount of work it's doing. To address this, choose a bigger EC2 instance type or request more resources if you're using Fargate.

Make sure to check out the documentation for [AWS CloudWatch Logs](#) and [AWS CloudWatch metrics](#).

Was this helpful?

✓ YES

✗ NO

Questions? [Contact us](#)

