Web Client for AWS Transfer Family

Implementation Guide

Web Client for AWS Transfer Family: Implementation Guide

Table of Contents

Provide a web portal for your users to access your corporate SFTP environments

Publication date: *October 2021*

The Web Client for AWS Transfer Family solution provides an intuitive web user interface for using AWS Transfer for Secure Shell File Transfer Protocol (SFTP). It allows you to adopt AWS Transfer Family plus provides a simple web portal to your corporate SFTP environments for your users.

Non-technical users find it inconvenient to use thick client applications, such as FileZilla and others to transfer files. It's also complicated to install and support different clients on various end user devices and operating systems. By adopting this browser-based solution you can avoid the effort of managing a commercial client and evade troubleshooting different end-user devices and operating systems. Your customers will be able to access your files without installing any software or using your system from the backend.

This solution supports common file operations, such as upload, download, rename and delete. Currently, the solution only supports AWS Transfer for SFTP.

This implementation guide describes architectural considerations and configuration steps for deploying Web Client for AWS Transfer Family in the AWS Cloud. It includes links to AWS CloudFormation templates that launch and configure the AWS services required to deploy this solution using AWS best practices for security and availability.

The guide is intended for IT infrastructure architects, administrators, and DevOps professionals who have practical experience architecting in the AWS Cloud.

Cost

You are responsible for the cost of the AWS services used while running this solution. As of October 2021, the cost for running this solution with the default settings in the US East (N. Virginia) Region is approximately **$207 per month**. This estimate excludes the cost associated with AWS Transfer for SFTP.

AWS service	Dimensions	Cost per month
Amazon S3	$0.023 per GB for first 50 TB / Month (Storage for the solution's web application and access logs only. Backend storage for AWS Transfer Family data not included.)	$0.023
Amazon CloudFront	Free tier includes 50 GB of data transfer out and 2,000,000 HTTPS requests each month for one year	$0.00
AWS Fargate	$0.01283328 per vCPU per hour and $0.00140919 per GB per hour **Note** Solution deploys 3 tasks, each using 0.25 vCPU and 0.5 GB	$7.69
Amazon DynamoDB	$1.25 per million write request units and $0.25 per million read request units **Note** First 25 GB stored per month is free. Solution does not expect to exceed few hundred KBs	$2.00
Interface Endpoint	$0.01 per VPC endpoint per AZ **Note** Solution deploys 8 endpoints across 3 AZs	$172.80
Application Load Balancer	$0.0225 per Application Load Balancer-hour (or partial hour) $0.008 per LCU-hour (or partial hour)	$21.96

AWS service	Dimensions	Cost per month
AWS Lambda	$0.20 per 1M requests and $0.0000000021 per 1ms for 128 MB configuration **Note** Solution deploys AWS Lambda function with 128 MB memory. In our experience, each invocation takes roughly 600ms and 1M invocations in a month	$3.00
Amazon API Gateway	Free tier	$0.00
	Total cost per month:	$207.00

We recommend creating a budget through AWS Cost Explorer to help manage costs. Prices are subject to change. For full details, refer to the pricing webpage for each AWS service used in this solution.

Architecture overview

Deploying this solution with the default parameters builds the following environment in the AWS Cloud.

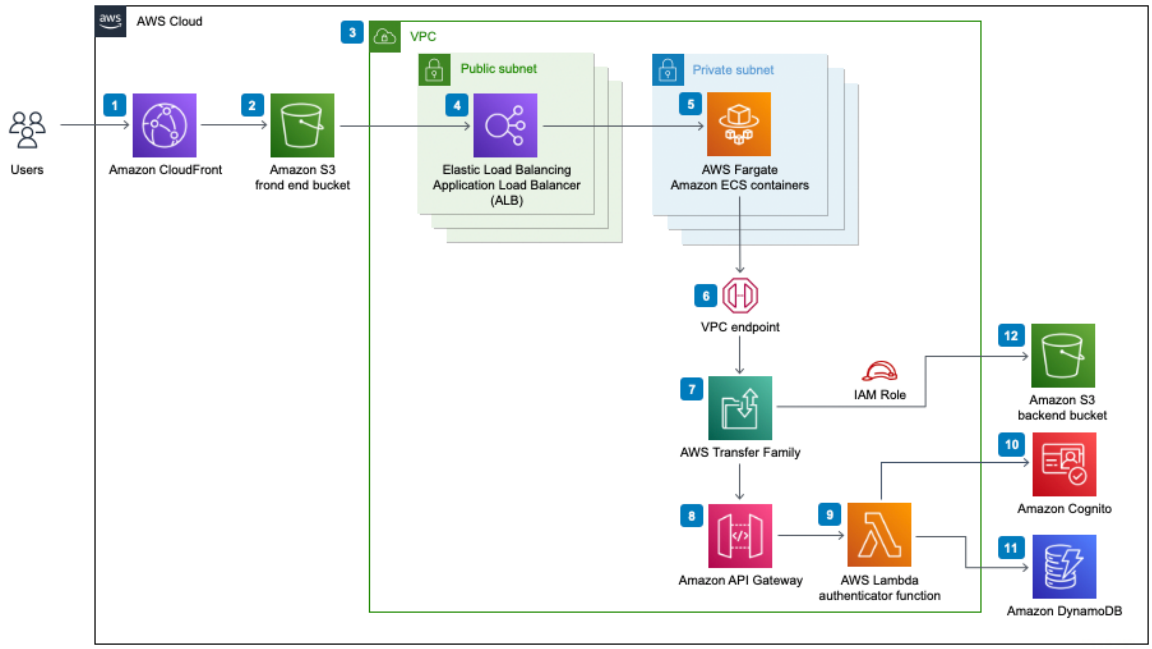

Figure 1: Web Client for AWS Transfer Family architecture on AWS

The AWS CloudFormation templates deploy the following infrastructure:

1. An Amazon CloudFront distribution to serve the solution's web application. Users interact with the Angular Single Page Application (SPA) front end after providing their Amazon Cognito credentials to log in.
2. An Amazon Simple Storage Service (Amazon S3) bucket configured for static website hosting.
3. A dedicated Amazon Virtual Private Cloud (Amazon VPC) with three private and three public subnets spread across three availability zones.
4. An Elastic Load Balancing Application Load Balancer that supports APIs for all file and folder operations.
5. Elastic Container Service (Amazon ECS) containers running on AWS Fargate in the three private subnets. The containers host a python-based application that exposes an API to front-end requests (log in, log out, list files and directories, create folders, upload, download, delete, and rename files).
6. VPC Endpoints for secure access to various services from Fargate containers. Fargate tasks programmatically interact with an AWS Transfer Family endpoint, making the SFTP connection using user-provided Amazon Cognito credentials.
7. An AWS Transfer Family SFTP server to provide an SFTP endpoint for file transfers. Transfer Family uses AWS IAM roles to integrate with the S3 bucket.
8. An Amazon API Gateway API to query Amazon Cognito and validate the end user's credentials.
9. An AWS Lambda function to support the API Gateway with authentication.
10. An Amazon Cognito user pool to manage user access to the web application and for custom authentication with AWS Transfer Family.

11 Amazon DynamoDB is used to store logical directory path mapping for AWS Transfer for SFTP server. The Lambda function also queries a DynamoDB table for the user's details, such as logical mapping.

12 An Amazon S3 bucket for storing the data for the AWS Transfer for SFTP server.

Solution components

The Web Client for AWS Transfer Family on AWS solution consists of two high-level components, a front end and a backend.

Web application

The front end consists of a single-page web application, deployed in S3 and restricted by Amazon CloudFront. Users of the solution access the web application over HTTPS and log in using their Amazon Cognito credentials to access their files and directories.

The web application is designed in Angular for users to perform required file operations (Upload, Download, Delete).

Backend

The backend for this solution is a Python Flask application, which runs on AWS Fargate behind a load balancer. The python application receives file operation requests from the front end and uses Paramiko library to transform them into SFTP operations, which are run against AWS Transfer Family's SFTP server.

Authentication

The user provides an Amazon Cognito username and password on initial login. During the login process, the user's credentials are programmatically authenticated against the AWS Transfer endpoint. Once the user is authenticated, the python application creates access and refresh JWT tokens using Flask's native framework and stores the user's username and password in the tokens after encrypting them using KMS Encrypt API. The JWT tokens are set in the access cookie (default name is set to **access_token_cookie**) and refresh cookie (default name is set to **refresh_token_cookie**) respectively. The tokens are used in subsequent HTTPS requests from the client to the web application where the python backend decrypts JWT tokens using KMS Decrypt API to get the username and password for that request and authenticate the request.

Security

When you build systems on AWS infrastructure, security responsibilities are shared between you and AWS. This shared model reduces your operational burden because AWS operates, manages, and controls the components including the host operating system, the virtualization layer, and the physical security of the facilities in which the services operate. For more information about AWS security, visit AWS Cloud Security.

IAM roles

AWS Identity and Access Management (IAM) roles allow customers to assign granular access policies and permissions to services and users on the AWS Cloud.

Amazon CloudFront

This solution deploys a web console hosted in an Amazon Simple Storage Service (Amazon S3) bucket. To help reduce latency and improve security, this solution includes an Amazon CloudFront distribution with an origin access identity, which is a CloudFront user that provides public access to the solution's website bucket contents. For more information, refer to Restricting Access to Amazon S3 Content by Using an Origin Access Identity in the *Amazon CloudFront Developer Guide*.

Security groups

The security groups created in this solution are designed to control and isolate network traffic. We recommend that you review the security groups and further restrict access as needed once the deployment is up and running.

Default throttling on API layer

The custom authentication Lambda function is fronted by API Gateway, which has default throttling of 10K requests per second and burst of 5K requests. Adjust these values per your requirement.

Design considerations

Replacing Amazon Cognito with another Identity Provider (IdP)

Even though this solution deploys and uses Amazon Cognito for user authentication and management with AWS Transfer service, it can be replaced with another custom IdP using API Gateway.

Deploying CloudFormation Templates

There are eight CloudFormation templates that are used to deploy this solution in your AWS account. The templates are numbered in the order to be deployed.

Regional deployments

This solution uses the AWS Fargate, AWS Transfer Family, and Amazon Cognito services, which are not currently available in all AWS Regions. You must launch this solution in an AWS Region where these services are available. For the most current availability by Region, refer to the AWS Regional Services List.

Supported AWS Regions

As of October 2021, this solution is supported in the following AWS Regions:

• us-east-1 (Virginia) • us-east-2 (Ohio) • us-west-2 (Oregon) • ap-south-1 (Mumbai) • ap-northeast-2 (Seoul) • ap-southeast-1 (Singapore) • ap-southeast-2 (Sydney) • ap-northeast-1 (Tokyo)	• ca-central-1 (Canada) • eu-central-1 (Frankfurt) • eu-west-1 (Ireland) • eu-west-2 (London) • eu-west-3 (Paris) • eu-north-1 (Stockholm) • me-south-1 (Bahrain) • sa-east-1 (Sao Paulo)

AWS CloudFormation templates

To automate deployment, this solution uses the following AWS CloudFormation templates, which you can download before deployment:

01-sftp-vpc.template: Use this template to launch Amazon VPC spanning three AZs. The default configuration deploys the required VPC and other best practices related resources such as security groups and VPC endpoints. You can customize the template based on your specific needs.

02-sftp-cognito.template: Use this template to create Amazon Cognito User Pool and User Pool Client. The default configuration deploys a Cognito user pool for provisioning users. You can customize the template based on your specific needs.

03-sftp-endpoint.template: Use this template to create the AWS Transfer Family SFTP server, Amazon API Gateway and Lambda function for custom authentication, AWS DynamoDB table for AWS Transfer user mapping and AWS Backup resources. The default configuration deploys an AWS Transfer Family SFTP server along with the required resources to support a working endpoint according to best practices. You can customize the template based on your specific needs.

04-sftp-ecs.template: Use this template to create the Amazon Container Service (Amazon ECS) infrastructure to host the containerized backend application. The default configuration creates an ECS cluster and related resources, such as an Amazon Elastic Container Registry (Amazon ECR) private registry to support the backend application. You can customize the template based on your specific needs.

05-sftp-fargate.template: Use this template to launch the AWS Fargate service representing the backend application. The default configuration deploys AWS Systems Manager Parameters stored in Parameter Store, AWS S3 buckets, and Amazon ECS task definition. You can also customize the template based on your specific needs.

06b-security-headers-lambda-edge.template: Use this template to implement security headers using Lambda@Edge.

07-sftp-web-client.template: Use this template to configure the user interface for the Web Client Solution. This stack creates the required AWS S3 buckets and Amazon CloudFront distribution to host a static website deployment on S3 fronted by CloudFront. You can customize the template based on your specific needs.

08-operational-metrics.template: Use this template to configure to collect and send operational metrics of the deployment of this solution.

Automated deployment

Before you launch the solution, review the cost, architecture, network security, and other considerations discussed in this guide. Follow the step-by-step instructions in this section to configure and deploy the solution into your account.

Time to deploy: Approximately 60 minutes

Deployment overview

Use the following steps to deploy this solution on AWS. For detailed instructions, follow the links for each step.

Prerequisites

- Set up an Amazon S3 bucket.
- Retrieve the ARN for your AWS Certificate Manager (ACM) certificate.
- Create an Amazon Route 53 hosted zone.
- Configure Node.js/Angular development environment.
- Clone the project code repository
- Install the Docker client.
- Create *dist* folder

Step 1. Launch the first four stacks (p. 13)

- Deploy the Amazon VPC stack.
- Deploy the Amazon Cognito stack.
- Deploy the SFTP endpoint stack.
- Deploy the Amazon ECS cluster stack.

Step 2. Create the application Docker image (p. 17)

Step 3. Launch the Fargate task stack (p. 19)

Step 4: Set up web client infrastructure (p. 20)

- Deploy and configure Lambda@Edge to insert security headers.
- Deploy the web client stack.
- Deploy web client

Step 5: Adding and mapping users (p. 24)

- Add an Amazon Cognito test user
- Configure DynamoDB table
- Log in and test the solution

Step 6: (Optional) Operational metrics stack deployment (p. 27)

Prerequisites

Set up Amazon S3 bucket

This solution uses an Amazon S3 bucket for storing the data for the AWS Transfer for SFTP server. Use prefixes to organize objects in the S3 bucket.

Retrieve ACM certificate ARN

This solution uses an ACM certificate to protect endpoints with your organization's own SSL certificate. Use the certificate's ARN when deploying the **05-sftp-fargate.template** CloudFormation template. To retrieve the ARN, refer to Listing certificates managed by ACM.

Create Route 53 hosted zone

This solution adds a record set specific to your SFTP endpoints. Verify that you have a Route 53 public hosted zone representing the domain for the solution's endpoints. Create a public hosted zone, if needed.

Configure Node.js/Angular

To build the solution's web-based user interface, you must configure a Node.js/Angular development environment on your local machine.

> **Note**
> This solution requires Node.js version 14.15 and Angular version 12 with related libraries.

1. In your terminal, install Node Version Manager (NVM):

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.34.0/install.sh | bash
```

2. Next, run nvm.sh file:

```
. ~/.nvm/nvm.sh
```

3. Install Node.js version 14.15:

```
nvm install 14.15
```

4. Make this version the default:

```
nvm alias default v14.15
```

5. Install Angular CLI:

```
npm install -g @angular/cli@12
```

Clone the project code repository

We use Amazon Linux Workspace as the workstation to deploy AWS CloudFormation templates. This provides a clean environment to install our dev dependencies without impact to other artifacts already installed on a desktop environment.

To clone the project code repository to your local workstation from the GitHub location, run the following command from your terminal:

```
git clone https://github.com/awslabs/web-client-for-aws-transfer-family
```

Install Docker client

Install Docker on your local machine to build the application's container image.

1. Install docker:

   ```
   sudo yum install docker
   ```

2. Start docker:

   ```
   sudo systemctl start docker
   ```

3. Start docker:

   ```
   sudo systemctl enable docker
   ```

4. Get status:

   ```
   sudo systemctl status docker
   ```

   If you experience DNS issues, such as: "Failed to establish a new connection: [Errno -3] Temporary failure in name resolution":

   a. Modify the docker daemon file:

      ```
      sudo vi /etc/docker/daemon.json
      ```

   b. Add the following content:

      ```
      {
           "dns": ["8.8.8.8", "8.8.4.4"]
       }
      ```

   c. Restart the docker service:

      ```
      sudo service docker restart
      ```

Create dist folder

Run the following script from the **deployment** folder. It creates a **dist** folder under the root of the solution folder.

```
./build-dist.sh solutions web-client-for-aws-transfer-family v1.0.0
```

(Optional) Customize UI icon and text

You can change the icon and text that appears on the top left in the web client UI.

1. From your local clone, in your file browser, navigate to the images path: **dist source frontend src assets images**.
2. Replace **logo.png** with the logo file of your choice.

You can change the text that appears on the top left in the web client UI.

1. From your local clone, in your file browser, navigate to the config path: **dist source frontend src assets config**.
2. Modify the `config.json` file. Use the following example to modify the `company_name`, `tagline`, and `site_title` attributes in the `config.json` file:

```
{
      "company_name": {
            "name": "Octank SFTP"
            },
            "tagline": {                  "tagline": "Web Client for AWS Transfer"

      },
      "site_title": {
"title": "Web Client for AWS Transfer"
}
}
```

Step 1. Launch the first four stacks

Several AWS CloudFormation templates deploy Web Client for AWS Transfer Family in the AWS Cloud. You must complete all aforementioned prerequisites before launching these stacks.

The first four stacks deploy Amazon VPC, Amazon Cognito, the SFTP endpoint, and Amazon ECS resources. Some of the resources from the earlier stacks are required values in later stack parameters. For example, the Amazon Cognito stack requires the Amazon VPC stack name. The default stack name parameter values in certain templates follow our example names. If you use different names for your stacks, use these consistently throughout the deployment of all stacks.

Note
You are responsible for the cost of the AWS services used while running this solution. For more details, visit the Cost (p. 2) section in this guide, and refer to the pricing webpage for each AWS service used in this solution.

Amazon VPC stack deployment

1. Sign in to the AWS CloudFormation console, select your Region for deployment, and choose **Create stack**, **With new resources (standard)**.

   Note
   This solution uses the AWS Fargate, AWS Transfer Family, and Amazon Cognito services, which are not currently available in all AWS Region. You must launch this solution in an AWS Region where these services are available. For the most current availability by Region, refer to the AWS Regional Services List.
2. On the **Specify template** page, under **Prepare template**, select **Template is ready**.
3. Under **Specify template**, select **Upload a template file**, then choose **Choose File** to upload the `01-sftp-vpc.template` AWS CloudFormation template from your local clone of the project under your **dist/deployment** folder**.** Choose **Next**.

4. On the **Specify stack details** page, assign a name to your solution stack, for example, *sftp-vpc-stack*. For information about naming character limitations, refer to IAM and STS Limits in the *AWS Identity and Access Management User Guide*.
5. Under **Parameters**, review the parameters for this solution template and modify them as necessary. This solution uses the following default values.

Parameter	Default	Description
PrivateSubnet1CIDR	`10.194.2.0/24`	The CIDR range for the first of the three private subnets.
PrivateSubnet2CIDR	`10.194.4.0/24`	The CIDR range for the second of the three private subnets.
PrivateSubnet3CIDR	`10.194.6.0/24`	The CIDR range for the third of the three private subnets.
PublicSubnet1CIDR	`10.194.1.0/24`	The CIDR range for the first of the three public subnets.
PublicSubnet2CIDR	`10.194.3.0/24`	The CIDR range for the second of the three public subnets.
PublicSubnet3CIDR	`10.194.5.0/24`	The CIDR range for the third of the three public subnets.
ResourceTag	`SFTPVPC`	Value to uniquely identify resources for this stack.
VpcCIDR	`10.194.0.0/21`	The VPC CIDR range.

6. Choose **Next**.
7. On the **Configure stack options** page, choose **Next**.
8. On the **Review** page, review and confirm the settings. Check the box acknowledging that the template will create AWS Identity and Access Management (IAM) resources.
9. Choose **Create stack** to deploy the stack.

   You can view the status of the stack in the AWS CloudFormation Console in the **Status** column. You should receive a CREATE_COMPLETE status in approximately four minutes.
10. From the CloudFormation Stacks page, choose the **Outputs** tab and record the values for **DefaultSecurityGroup**. Use the **DefaultSecurityGroup** value for deploying the SFTP endpoint stack.

Amazon Cognito stack deployment

1. Sign in to the AWS CloudFormation console, select your Region for deployment, and choose **Create stack**, **With new resources (standard)**.

   Important
   Launch the stack in the same Region that you selected for the VPC stack.
2. On the **Specify template** page, under **Prepare template**, select **Template is ready**.
3. Under **Specify template**, select **Upload a template file**, then choose **Choose File** to upload the `02-sftp-cognito.template` AWS CloudFormation template from your local clone of the project under your **dist/deployment** folder**.** Choose **Next**.

4. On the **Specify stack details** page, assign a name to your solution stack, for example, *sftp-cognito-stack*. For information about naming character limitations, refer to IAM and STS Limits in the *AWS Identity and Access Management User Guide*.
5. Under **Parameters**, review the parameters for this solution template and modify them as necessary. This solution uses the following default value.

Parameter	Default	Description
VPCResourceStack	`sftp-vpc-stack`	Cross stack reference to the stack used to deploy the VPC resources. Use the name of the VPC stack from Amazon VPC stack deployment (p. 13), step 5.

6. Choose **Next**.
7. On the **Configure stack options** page, choose **Next**.
8. On the **Review** page, review and confirm the settings.
9. Choose **Create stack** to deploy the stack.

   You can view the status of the stack in the AWS CloudFormation Console in the **Status** column. You should receive a CREATE_COMPLETE status in approximately one minute.
10. From the CloudFormation Stacks page, choose the **Outputs** tab and record the values for **UserPoolClientId** and **UserPoolId**. These are for adding and mapping users (p. 24) to the solution.

SFTP endpoint stack deployment

1. Sign in to the AWS CloudFormation console, select your Region for deployment, and choose **Create stack**, **With new resources (standard)**.

   Important
   Launch the stack in the same Region that you selected for the VPC stack.
2. On the **Specify template** page, under **Prepare template**, select **Template is ready**.
3. Under **Specify template**, select **Upload a template file**, then choose **Choose File** to upload the `03-sftp-endpoint.template` AWS CloudFormation template from your local clone of the project under your **dist/deployment** folder**.** Choose **Next**.
4. On the **Specify stack details** page, assign a name to your solution stack, for example, *sftp-endpoint-stack*. For information about naming character limitations, refer to IAM and STS Limits in the *AWS Identity and Access Management User Guide*.
5. Under **Parameters**, review the parameters for this solution template and modify them as necessary. This solution uses the following default values.

Parameter	Default	Description
AWSTransferForSFTPS3Bucket	*<Requires input>*	The Amazon S3 bucket that stores the data for the AWS Transfer Family server (backend for AWS Transfer Family endpoint). Note: This S3 bucket is not created as part of this stack, you must create the bucket beforehand. For details, refer to Prerequisites (p. 11).

Parameter	Default	Description
AWSTransferVPCSecGroup	*<Requires input>*	The default security group created as part of the 01-sftp-vpc.template stack. Use the output value of **DefaultSecurityGroup** from Amazon VPC stack deployment (p. 13), step 11.
CognitoResourceStack	`sftp-cognito-stack`	Cross stack reference to the stack used to deploy the Cognito resources (02-sftp-cognito.template). Use the name of the Cognito stack from Amazon Cognito stack deployment (p. 14), step 5.
VPCResourceStack	`sftp-vpc-stack`	Cross stack reference to the stack used to deploy the VPC resources. Use the name of the VPC stack from Amazon VPC stack deployment (p. 13), step 5.

6. Choose **Next**.
7. On the **Configure stack options** page, choose **Next**.
8. On the **Review** page, review and confirm the settings. Check the box acknowledging that the template will create AWS Identity and Access Management (IAM) resources.
9. Choose **Create stack** to deploy the stack.

   You can view the status of the stack in the AWS CloudFormation Console in the **Status** column. You should receive a CREATE_COMPLETE status in approximately 12 minutes.

Amazon ECS cluster stack deployment

1. Sign in to the AWS CloudFormation console, select your Region for deployment, and choose **Create stack**, **With new resources (standard)**.

   Important
   Launch the stack in the same Region that you selected for the VPC stack.
2. On the **Specify template** page, under **Prepare template**, select **Template is ready**.
3. Under **Specify template**, select **Upload a template file**, then choose **Choose File** to upload the `04-sftp-ecs.template` AWS CloudFormation template from your local clone of the project under your **dist/deployment** folder**.** Choose **Next**.
4. On the **Specify stack details** page, assign a name to your solution stack, for example, *sftp-ecs-stack*. For information about naming character limitations, refer to IAM and STS Limits in the *AWS Identity and Access Management User Guide*.
5. Under **Parameters**, review the parameters for this solution template and modify them as necessary. This solution uses the following default values.

Parameter	Default	Description
HostedZoneId	*<Requires input>*	The ID of the Route 53 hosted zone you are adding a record set to pointing to your backend API URL (the Application Load Balancer fronting the backend API).
RecordName	`sftpapi.mycompanydomain.com`	The Domain Name System (DNS) name for a new record set. For ex ample, if the hosted zone is *mycompanydomain.com*, then enter *<RecordName>*`.mycompanydomain.com`. This record set points to the Application Load Balancer fronting the backend API on Amazon ECS.
VPCResourceStack	`sftp-vpc-stack`	Cross stack reference to the stack used to deploy the VPC resources. Use the name of the VPC stack from Amazon VPC stack deployment (p. 13), step 5.

6. Choose **Next**.
7. On the **Configure stack options** page, choose **Next**.
8. On the **Review** page, review and confirm the settings. Check the box acknowledging that the template will create AWS Identity and Access Management (IAM) resources.
9. Choose **Create stack** to deploy the stack.

   You can view the status of the stack in the AWS CloudFormation Console in the **Status** column. You should receive a CREATE_COMPLETE status in approximately three minutes.
10. From the CloudFormation Stacks page, choose the **Outputs** tab and record the value for **ECR**. Use this value when creating the application Docker image.

Step 2: Create the application Docker image

After deploying the first four stacks, build the container image representing the SFTP API backend application. Next, deploy this image to Amazon ECS.

> **Note**
> To successfully create the application Docker image, the script requires that you use an AWS_PROFILE with sufficient privileges to run the scripted API commands.

1. From your local clone of the project, under **dist/deployment** folder, select the **04a-build-docker-image.sh** file and modify it in your text editor.

   > **Note**
   > You can also modify the file in the terminal using nano or vi editor.

   Use the following sample to modify the `COMPANY_DOMAIN`, `CLOUDFRONT_CNAME`, and `ECR_REPO_NAME` variables in the `04a-build-docker-image.sh` file:

- Update `COMPANY_DOMAIN` (line 10) to your corporate domain, for example, *mycompanydomain.com*.
- Update `CLOUDFRONT_CNAME` (line 11) to the CNAME for your CloudFront distribution, for example, *ui.mycompanydomain.com*.
- Update `ECR_REPO_NAME` (line 12) to the value of **ECR** key from Amazon ECS cluster stack deployment (p. 16) step 11, for example, *sftp-backend-0a635743*.

```
# Setup env vars
AWS_PROFILE=$1

ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text --profile
 $AWS_PROFILE)

# INSTRUCTIONS - update with target region
AWS_REGION=$(aws configure get region --profile $AWS_PROFILE)

# Replace with your R53 Alias pointing to your ALB
COMPANY_DOMAIN='<enter_your_domain>'        # example mycompanydomain.com
CLOUDFRONT_CNAME='<enter_cloudfront_cname>' # example ui.mycompanydomain.com
ECR_REPO_NAME='<enter_ecr_repo_name>' # example sftp-backend-0a635743

# Build the image from the source directory
pushd ../source/backend

sed -i -e "s/REPLACE_ME_COMPANY_DOMAIN/$COMPANY_DOMAIN/" src/
flask_app_jwt_configuration.json
sed -i -e "s/REPLACE_ME_CLOUDFRONT_CNAME/$CLOUDFRONT_CNAME/" src/
flask_app_jwt_configuration.json

sudo docker build -t sftp-backend .

popd

sed -i -e "s/REPLACE_ME_COMPANY_DOMAIN/$COMPANY_DOMAIN/" 06a-add-security-headers.sh

# Log in to ECR and push image
aws ecr get-login-password \
  --region $AWS_REGION \
  --profile $AWS_PROFILE | sudo docker login --username AWS \
  --password-stdin "$ACCOUNT_ID.dkr.ecr.$AWS_REGION.amazonaws.com"

sudo docker tag sftp-backend "$ACCOUNT_ID.dkr.ecr.$AWS_REGION.amazonaws.com/
$ECR_REPO_NAME:latest"

sudo docker push "$ACCOUNT_ID.dkr.ecr.$AWS_REGION.amazonaws.com/$ECR_REPO_NAME:latest
```

2. From your terminal, run the following script:

```
./04a-build-docker-image.sh <AWS_PROFILE>
```

Note
To adjust the permission on the file, use chmod command to make it runnable. Use sudo if you experience any permission issues after trying chmod.

3. Sign in to the AWS Elastic Container Registry console.
4. Under **Repositories**, select the ECR repository recorded in Amazon ECS cluster stack deployment (p. 16), step 11 and copy the value for **Image URI**. Use the **ImageURI** value for deploying the Fargate task stack.

Step 3: Launch the Fargate task stack

1. Sign in to the AWS CloudFormation console, select your Region for deployment, and choose **Create stack**, **With new resources (standard)**.

   Important
   Launch the stack in the same Region that you selected for the VPC stack.
2. On the **Specify template** page, under **Prepare template**, select **Template is ready**.
3. Under **Specify template**, select **Upload a template file**, then choose **Choose File** to upload the `05-sftp-fargate.template` AWS CloudFormation template from your local clone of the project under your **dist/deployment** folder**.** Choose **Next**.
4. On the **Specify stack details** page, assign a name to your solution stack, for example, *sftp-fargate-stack*. For information about naming character limitations, refer to IAM and STS Limits in the *AWS Identity and Access Management User Guide*.
5. Under **Parameters**, review the parameters for this solution template and modify them as necessary. This solution uses the following default values.

Parameter	Default	Description
ACMCertificateARN	*<Requires input>*	The ARN of the ACM certificate used for HTTPS Application Load Balancer (ALB) endpoint. **Note** Use the ARN of the certificate defined within the same Region as the ALB. For details, refer to Prerequisites (p. 11).
ContainerCPU	`256`	Amount of CPU for each task; 1024 = 1 vCPU. For supported CPU and memory values for Fargate tasks, refer to Invalid CPU or memory value specified in the *User Guide for AWS Fargate*.
ContainerMemory	`512`	Memory in GB for each task.
ContainerPort	`80`	The `containerPort` that the container uses to receive traffic.
ECSStackName	`sftp-ecs-stack`	Cross stack reference to the stack used to deploy the Amazon ECS resources. Use the name of the ECS stack from Amazon ECS cluster stack deployment (p. 16), step 5.
ImageURI	*<Requires input>*	The ECR Image URI for the application. Use the output value of **ImageURI** from Create the application Docker image (p. 17), step 4.

Parameter	Default	Description
SFTPEndPointStackName	`sftp-endpoint-stack`	Cross stack reference to the stack used to deploy the SFTP resources. Use the name of the SFTP endpoint stack from SFTP endpoint stack deployment (p. 15), step 5.
TaskCount	`3`	Number of Fargate tasks to deploy. The default is three to provide one task for each of the three AZs.
VPCResourceStack	`sftp-vpc-stack`	Cross stack reference to the stack used to deploy the VPC resources. Use the name of the VPC stack from Amazon VPC stack deployment (p. 13), step 5.

6. Choose **Next**.
7. On the **Configure stack options** page, choose **Next**.
8. On the **Review** page, review and confirm the settings. Check the box acknowledging that the template will create AWS Identity and Access Management (IAM) resources.
9. Choose **Create stack** to deploy the stack.

   You can view the status of the stack in the AWS CloudFormation Console in the **Status** column. You should receive a CREATE_COMPLETE status in approximately one minute.
10. From the CloudFormation Stacks page, choose the **Outputs** tab and record the value for **SFTPWebClientBucket**. Use this value when setting up the web client infrastructure.

Step 4: Set up web client infrastructure

After deploying the Fargate task stack, you can configure the Lambda@Edge function that adds necessary security headers (Strict-Transport-Security, X-Frame-Options, X-XSS-Protection, X-Content-Type-Options, Content-Security-Policy), which are used by the CloudFront distribution.

> **Note**
> To successfully set up the web client infrastructure, the script requires that you pass an AWS_PROFILE with sufficient privileges to execute the scripted API commands.

1. From your local clone of the project, under the **dist/deployment** folder, select the **06-add-security-header.sh** file and modify it in your text editor.

   > **Note**
   > You can also modify the file in the terminal using nano or vi editor.
   > Use the following sample to modify the `BACKEND_URL` and `SFTP_WEB_CLIENT_BUCKET` variables in the `06-add-security-header.sh` file:

```
# Setup env vars
AWS_PROFILE=$1

# Replace with your preferred domain name
DOMAIN_NAME='REPLACE_ME_COMPANY_DOMAIN' # NOTE: example, 'mycompanydomain.com'

# Create Cloudformation stack that creates Lambda@Edge function in US-East-1
```

```
stack_name="sftp-sec-hdr-stack-$(echo $RANDOM)"
echo $stack_name

# This template needs to be deployed in US-EAST-1 as Lambda@Edge functions are currently
# required to be in US-EAST-1 region
aws cloudformation create-stack --stack-name $stack_name --template-
body file://06b-security-headers-lambda-edge.template --parameters
 ParameterKey=DomainName,ParameterValue=$DOMAIN_NAME \
 --profile $AWS_PROFILE --region us-east-1 --capabilities CAPABILITY_IAM

# Print the value of Lambda@Edge version, which will be need in Cloudfront distribution
 configuration next.
lambda_edge_version=$(aws cloudformation describe-stacks --stack-name $stack_name --
profile $AWS_PROFILE --region us-east-1 | grep OutputValue)
while [ -z "$lambda_edge_version" ] ; do # check for null
  lambda_edge_version=$(aws cloudformation describe-stacks --stack-name $stack_name --
profile $AWS_PROFILE --region us-east-1 | grep OutputValue)
  echo "Waiting for stack to complete its creation."
  sleep 10
done
echo $lambda_edge_version
```

2. From your terminal, run the following script:

```
./ 06-add-security-header.sh <AWS_PROFILE>
```

Note
To adjust the permission on the file, use chmod command to make it runnable.

3. From the CloudFormation Stacks page, choose the **Outputs** tab and record the value for **OutputValue**. This value is the Lambda@Edge function's version ARN required for deploying the web client stack.

Web client stack deployment

1. Sign in to the AWS CloudFormation console, select your Region for deployment, and choose **Create stack**, **With new resources (standard)**.

   Important
   Launch the stack in the same Region that you selected for the VPC stack.

2. On the **Specify template** page, under Prepare template, select **Template is ready**.
3. Under **Specify template**, select **Upload a template file**, then choose **Choose File** to upload the `07-sftp-web-client.template` AWS CloudFormation template from your local clone of the project under your **dist/deployment** folder**.** Choose **Next**.
4. On the **Specify stack details** page, assign a name to your solution stack, for example, *sftp-webclient-stack*. For information about naming character limitations, refer to IAM and STS Limits in the *AWS Identity and Access Management User Guide*.
5. Under **Parameters**, review the parameters for this solution template and modify them as necessary. This solution uses the following default values.

Parameter	Default	Description
ACMCertificateARN	`arn:aws:acm:us-east-1:111222333444:certificate/99988822-5a2b-4e8e-aaf0-83d8deed4445`	The ARN of the ACM certificate used for CloudFront. **Important** This certificate, unlike the certificate used for the ALB in the Fargate

Parameter	Default	Description
		task deployment, must be in us-east-1, regardless of what Region you choose for your solution deployment. For details, refer to Prerequisites (p. 11).
CNameAlternateDomainName	`ui.mycompanydomain.com`	CloudFront distribution alternate domain name that matches R53 domain name for your user interface.
FargateResourceStack	`sftp-fargate-stack`	Cross stack reference to the stack used to deploy the Fargate resources. Use the name of the Fargate task stack from Fargate task stack deployment (p. 19), step 5.
HostedZoneId	*<Requires input>*	The ID of the Route 53 hosted zone you are adding a record set pointing to your CloudFront distribution URL (the distribution serving your user interface). This is the same Route 53 ID used for the ECS stack deployment.
LambdaEdgeVersionARN	*<Requires input>*	ARN of the Lambda@Edge function version to be used in CloudFront distribution. Use the value of **OutputValue** from Set up web client architecture (p. 20), step 2.
VPCResourceStack	`sftp-vpc-stack`	Cross stack reference to the stack used to deploy the VPC resources. Use the name of the VPC stack from Amazon VPC stack deployment (p. 13), step 5.

6. Choose **Next**.
7. On the **Configure stack options** page, choose **Next**.
8. On the **Review** page, review and confirm the settings. Check the box acknowledging that the template will create AWS Identity and Access Management (IAM) resources.
9. Choose **Create stack** to deploy the stack.

   You can view the status of the stack in the AWS CloudFormation Console in the **Status** column. You should receive a CREATE_COMPLETE status in approximately five minutes.

Web client deployment

After deploying the web client stack, compile the web UI project and deploy the compiled assets to the S3 bucket designated to host the solution's static website. In the prior deployment step, template automatically sets up a CloudFront distribution that points to this S3 bucket as the origin for our user interface.

> **Note**
> To successfully set up the web client infrastructure, the script requires that you pass an AWS_PROFILE with sufficient privileges to execute the scripted API commands.

1. Open a terminal and navigate to the **source/frontend** folder:

```
cd source/frontend
```

2. Install the project dependencies:

```
npm install
```

3. Navigate to the **dist/deployment** folder:

```
cd dist/deployment
```

4. Select the **07a-sftp-web-assets-to-s3.sh** file and modify the `BACKEND_URL` and `SFTP_WEB_CLIENT_BUCKET` environment variables:
   - Update the `BACKEND_URL` value to reflect the Route 53 RecordSet pointing to your ALB. Refer to Amazon ECS cluster stack deployment (p. 16), Step 5, input parameter **RecordName**. Follow the formatting indicated in the file: *https:\/\/sftpapi.mycompanydomain.com*.
   - Update the `SFTP_WEB_CLIENT_BUCKET` value with the name of the bucket recorded in Launch the Fargate task stack (p. 19), step 10.

```
# Setup env vars
AWS_PROFILE=$1

# Replace with your R53 Alias pointing to your ALB
BACKEND_URL='REPLACE_ME' # NOTE: example, and make sure to use the following format
 `https:\/\/sftpapi.mycompanydomain.com`


# replace with s3 static website bucket created in sftp-web-client-stack
SFTP_WEB_CLIENT_BUCKET=REPLACE_ME # example sftp-web-ui-artifacts-f57a2400

# Build the image from the source directory
pushd ../source/frontend

aws s3 rm s3://$SFTP_WEB_CLIENT_BUCKET --recursive --profile $AWS_PROFILE

rm -rf ./dist

sed -i -e "s/REPLACE_ME/$BACKEND_URL/" src/app/service/ftp.service.ts

npm install

ng build --prod

aws s3 sync --cache-control 'max-age=604800' --exclude index.html dist/sftp-ng-webui
 s3://$SFTP_WEB_CLIENT_BUCKET --profile $AWS_PROFILE

# index.html is not stored with cach control headers because we want a unique
 index.html created on every deploy
```

```
aws s3 sync --cache-control 'no-cache' dist/sftp-ng-webui s3://$SFTP_WEB_CLIENT_BUCKET/
 --profile $AWS_PROFILE

popd
```

5. From your terminal, run the following script:

```
./07a-sftp-web-assets-to-s3.sh <AWS_PROFILE>
```

Note
To adjust the permission on the file, use chmod command to make it runnable.

Upon successful completion, the compiled assets will display in the S3 bucket.

Step 5. Adding and mapping users

Use the following procedure to allow users to log in and use the web client.

Add an Amazon Cognito user

1. Open a terminal and navigate to the **dist/deployment** folder:

```
cd dist/deployment
```

2. Edit the following parameters in the `07b-cognito-seed.sh` file:

Parameter	Description
COGNITO_CLIENT_ID	Use the output value of **UserPoolClientId** from Amazon Cognito stack deployment (p. 14), step 11.
COGNITO_APP_CLIENT_SECRET	For details on getting this value, refer to Amazon Cognito stack deployment (p. 14).
COGNITO_USER_USERNAME	Provide a username.
COGNITO_USER_PASSWD	Enter a password consisting of at least one upper case, lower case, number, and special character, for example, *TempPass@123456*.
COGNITO_USER_TELNUM	Example: *+123 123 1234* **Note:** This phone number can be a dummy number.
COGNITO_USER_EMAIL_DOMAIN	Example: *amazon.com*
COGNITO_USERPOOL_ID	Use the output value of **UserPoolId** from Amazon Cognito stack deployment (p. 14), step 11.

3. Run the following script:

```
./07b-cognito-seed.sh <AWS_PROFILE>
```

4. When the script completes, navigate to your Cognito User Pool to verify your test user was created. Sign in to the Amazon Cognito console.
5. Choose **Manage User Pools**, then select this solution's Amazon Cognito user pool.
6. Select **Users and groups**, and verify your test user appears in the list.

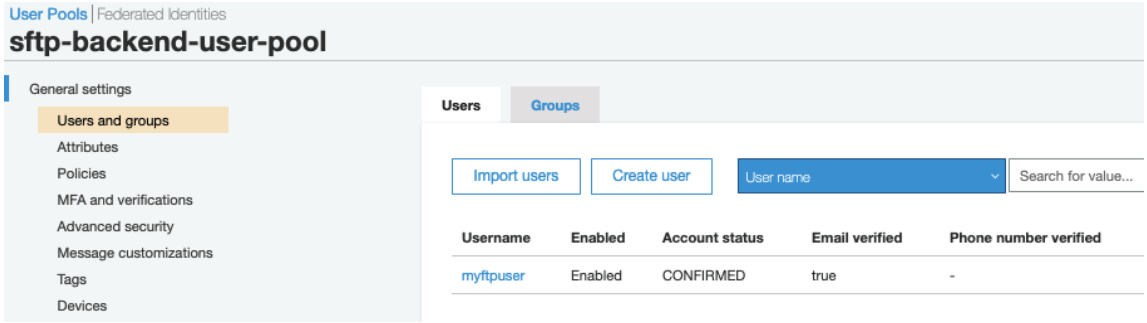

Figure 2: Amazon Cognito user pool

Configure DynamoDB table

Configure the mapping of the Amazon Cognito user created earlier to the Amazon S3 bucket designated for the SFTP backend. This bucket contains a folder for each user. In this example, the test user *mysftpuser* is mapped to the `sftp-backend-bucket` S3 bucket, thus mapping a folder from the designated bucket to this username. Here is the entry you must add to **SFTPUserDirectoryMapping** DynamoDB, creating by the CloudFormation templates in prior sections. The *target* attribute defines the path to the folders that the user should have access to. The *entry* attribute defines what the user will see when they log in to the web client.

```
{
      "directoryMap": [
            {
                  "Entry": "/team/sales",
                  "Target": "/sftp-backend-bucket-eu-west-1/sales"
            },
            {
                  "Entry": "/personal",
                  "Target": "/sftp-backend-bucket-eu-west-1/jdoe"
            }
      ],
      "username": "jdoe"
}
```

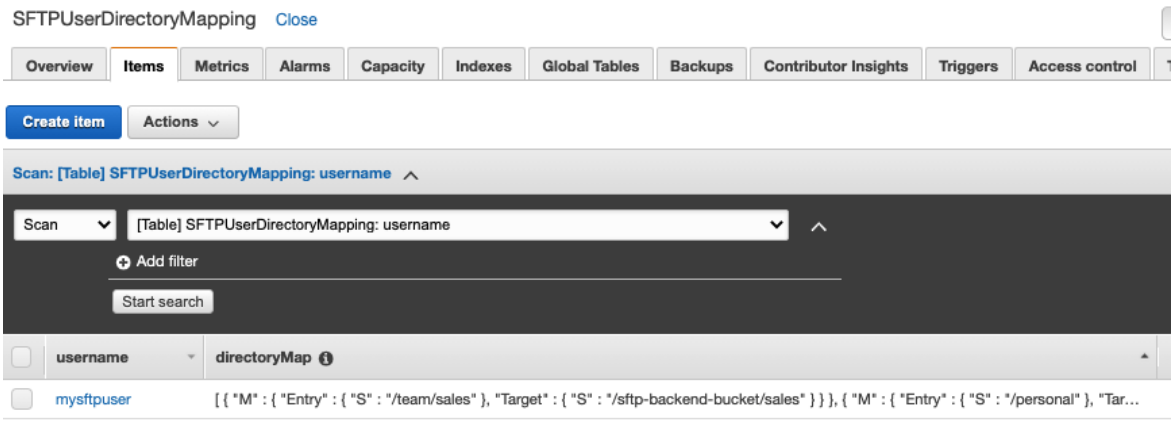

Figure 3: DynamoDB mapping of Amazon Cognito user

Log in and test the solution

1. Open a browser and navigate to CloudFront distribution alternate domain name entered during Web client stack deployment (p. 21), step 6.
2. Provide username, password of your user, and then choose **Log in**.

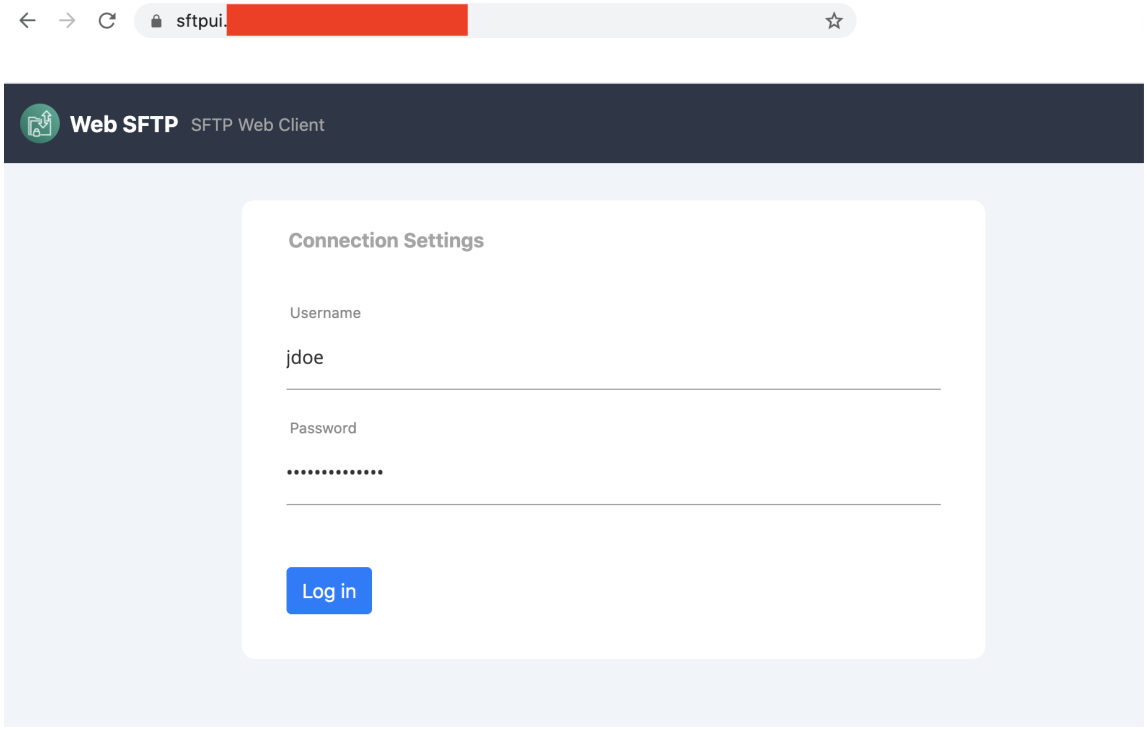

Figure 4: Web Client for AWS Transfer Family login page

Upon logging in, the user will have access to their permitted folders and documents.

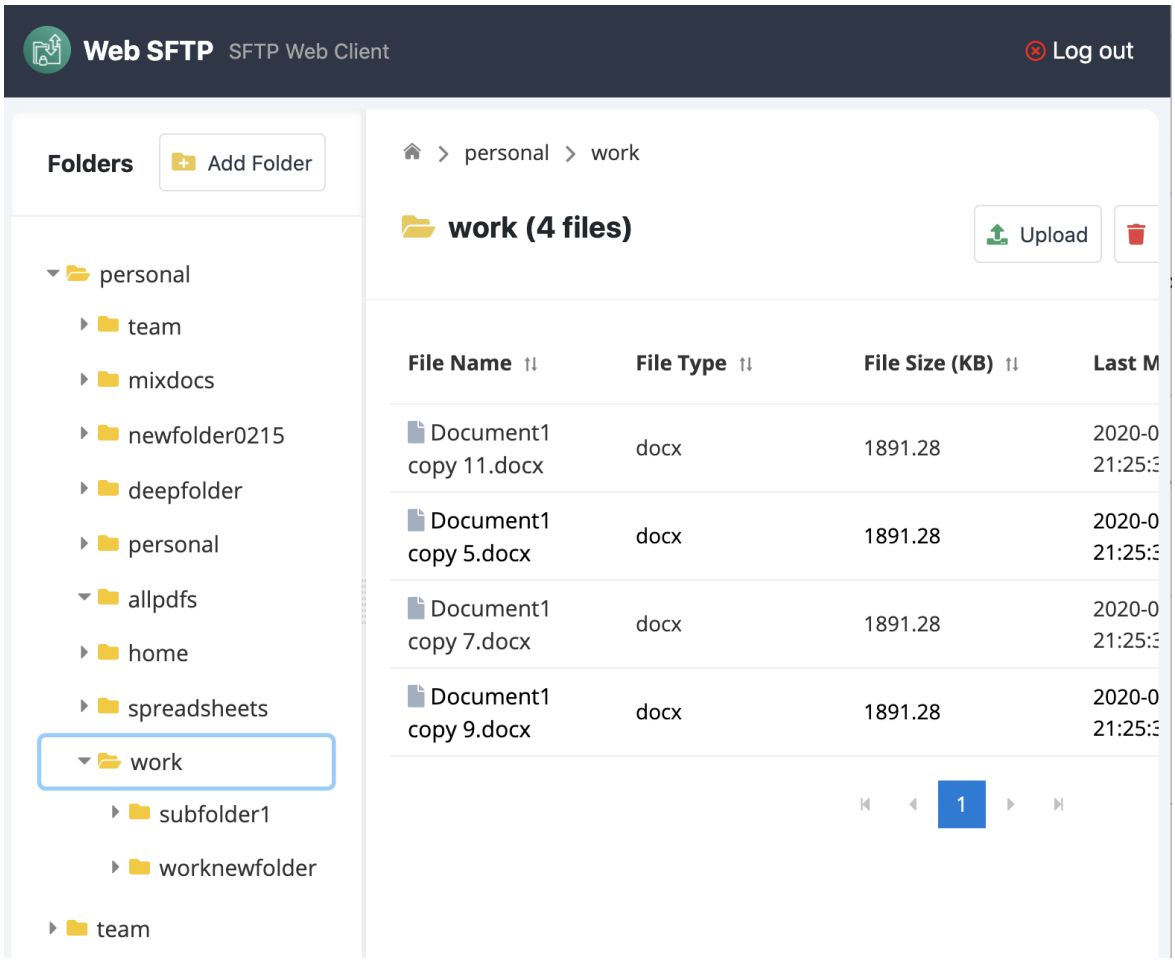

Figure 5: Solution's web application

> **Note**
> This solution does not preserve the source client IP in AWS Transfer CloudWatch Logs.

Step 6. (Optional) Operational metrics stack deployment

Under the **dist/deployment** folder, you can run the `08-operational-metrics.template` to help us gather operational metrics for this solution.

1. Sign in to the AWS CloudFormation console, select your Region for deployment, and choose **Create stack**, **With new resources (standard)**.

   > **Important**
   > Launch the stack in the same Region that you selected for the VPC stack.

2. On the **Specify template** page, under Prepare template, select **Template is ready**.
3. Under **Specify template**, select **Upload a template file**, then choose **Choose File** to upload the `08-operational-metrics.template` AWS CloudFormation template from your local clone of the project under your **dist/deployment** folder**.** Choose **Next**.

4. On the **Specify stack details** page, assign a name to your solution stack, for example, *sftp-operational-metrics-stack*. For information about naming character limitations, refer to IAM and STS Limits in the *AWS Identity and Access Management User Guide*.
5. Under **Parameters**, review the parameters for this solution template and modify them as necessary. This solution uses the following default value.

Parameter	Default	Description
CognitoStackName	`sftp-cognito-stack`	Cross stack reference to the stack used to deploy the Cognito resources (`02-sftp-cognito.template`). Use the name of the Cognito stack from Amazon Cognito stack deployment (p. 14), step 5.
ECSStackName	`sftp-ecs-stack`	Cross stack reference to the stack used to deploy the Amazon ECS resources. Use the name of the ECS stack from Amazon ECS cluster stack deployment (p. 16), step 5.
SendAnonymousUsage	`Yes`	Send operational metrics of this solution anonymously.
VPCResourceStack	`sftp-vpc-stack`	Cross stack reference to the stack used to deploy the VPC resources. Use the name of the VPC stack from Amazon VPC stack deployment (p. 13), step 5.

6. Choose **Next**.
7. On the **Configure stack options** page, choose **Next**.
8. On the **Review** page, review and confirm the settings. Check the box acknowledging that the template will create AWS Identity and Access Management (IAM) resources.
9. Choose **Create stack** to deploy the stack.

   You can view the status of the stack in the AWS CloudFormation Console in the **Status** column. You should receive a CREATE_COMPLETE status in approximately five minutes.

Using the solution

Supported file and folder operations

The solution supports the following file and folder operations:

Operation	File level	Folder level	Notes
List	Yes	Yes	
Upload	Yes	No	Only one file at a time. File drag and drop is also supported.
Download	Yes	No	Only a single file at a time can be downloaded.
Rename	Yes	No	
Delete	Yes	Yes	While deleting a folder, it should be empty, otherwise nothing will happen.
Create	No	Yes	

Uploading and downloading large files

The Gunicorn connection timeout is set to 600 seconds for sync workers. To upload and download files of large sizes, you can adjust the timeout value to be set at a higher interval.

Modify line 43 of the Dockerfile from your local clone of the project (under **source** > **backend** > **Dockerfile**):

```
ENTRYPOINT gunicorn --bind 0.0.0.0:80 transfer_sftp_backend:app --timeout 600
```

Monitoring and logging

The CloudWatch logs are available under **/ecs/sftp-ecs-logs** log group. The logging convention is as follows:

Class | LogLevel | FargateTaskID(GunicornProcessID) – OperationName - Messsage

Adjust the log level by changing the configuration in **dist** > **source** > **backend** > **src** > **logging.conf** file by changing the following attribute:

```
level=INFO
```

In addition, the provided CloudWatch logs, you can activate Container Insights on your AWS Fargate cluster by referencing Amazon ECS CloudWatch Container Insights in the *User Guide for AWS Fargate*. This will give you performance insights (CPU, Memory, Network, Storage) into your Fargate tasks.

The Upload and Download operations use the Fargate task's ephemeral storage. The solution also deploys **Fargate Task Ephemeral Storage**, a custom ECS CloudWatch metric, which shows the amount of ephemeral storage remaining for specific Fargate tasks.

Additional resources

AWS services	
• AWS CloudFormation	• AWS Transfer Family
• Amazon Cognito	• AWS Identity and Access Management (IAM)
• Amazon Simple Storage Service (Amazon S3)	• Amazon API Gateway
• Amazon Virtual Private Cloud (Amazon VPC)	• AWS Lambda
• Elastic Load Balancing	• Amazon DynamoDB
• Amazon Elastics Container Service	• Amazon CloudFront
• AWS Fargate	• Amazon Route 53

Uninstall the solution

You can uninstall the Web Client for AWS Transfer Family solution from the AWS Management Console. You must manually empty the Amazon S3 buckets, ECR repository, and AWS Backup vaults used by this solution.

Using the AWS Management Console

1. Sign in to Sign in to the Amazon S3 console and empty the following S3 buckets:
   - cloudfront-logs-*<stackID>*
   - sftp-web-ui-artifacts-*<stackID>*
   - sftp-web-ui-access-logs-*<stackID>*
2. Sign in to the ECR console and delete the sftp-backend ECR repository in the Region of your deployment.
3. Sign in to the EC2 console.
4. In the left navigation, select **Load Balancers** and select the ALB created as part of the deployment. Ensure **Delete Protection** is disabled on the public load balancer.
5. Sign in to the AWS Backup console.
   a. In the left navigation, select **Backup vaults**, select the `BackupVaultWithDailyBackups-`*<stackID>* backup vault and delete all recovery points.
   b. Delete the `DynamoDBBackupVaultWithDailyBackups-`*<stackID>* backup vault.
6. Sign in to the AWS CloudFormation console. On the **Stacks** page, select the following stacks and choose **Delete**:
   - sftp-webclient-stack
   - sftp-fargate-stack
   - sftp-ecs-stack
   - sftp-endpoint-stack
   - sftp-cognito-stack
   - sftp-vpc-stack

Collection of operational metrics

This solution includes an option to send anonymous operational metrics to AWS. We use this data to better understand how customers use this solution and related services and products. When invoked, the following information is collected and sent to AWS:

- **Solution ID:** The AWS solution identifier
- **Unique ID (UUID):** Randomly generated, unique identifier for each Web Client for AWS Transfer Family deployment
- **Timestamp:** Data-collection timestamp

AWS owns the data gathered though this survey. Data collection is subject to the AWS Privacy Policy. To opt out of this feature, complete the following steps before launching the AWS CloudFormation template.

To opt out of this feature, set the **SendAnonymousUsage** parameter in the solution's `08-operational-metrics.template` AWS CloudFormation template to a value of false. For details, refer to Operational metrics stack deployment (p. 27).

Source code

Visit our GitHub repository to download the source files for this solution and to share your customizations with others. Refer to the README.md file for additional information.

Revisions

Date	Change
October 2021	Initial release

Contributors

- Kapil Shardha
- Juan Lamadrid
- Aravind Singirikonda

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents AWS current product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. AWS responsibilities and liabilities to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

Web Client for AWS Transfer Family is licensed under the terms of the of the Apache License Version 2.0 available at The Apache Software Foundation.

AWS glossary

For the latest AWS terminology, see the AWS glossary in the *AWS General Reference*.