AKIAJAOG573TNAAMBGHQ

p2+noEiRnZPWicu/fEScsbaAfqGRkoz7ua5Nx9L5

**What Is AWS Step Functions?**

Amazon Simple Queue Service (SQS) is a fully managed message queuing service that enables you to decouple and scale microservices, distributed systems, and serverless applications.

Step Functions provides a graphical console to visualize the components of your application as a series of steps. It automatically triggers and tracks each step, and retries when there are errors, so your application executes in order and as expected, every time.

## Supported Regions

Currently, Step Functions is supported only in the following regions:

* US East (Ohio)
* US East (N. Virginia)
* US West (Oregon)
* US West (N. California)
* Asia Pacific (Mumbai)
* China (Ningxia)
* China (Beijing)
* Asia Pacific (Sydney)
* Asia Pacific (Tokyo)
* Asia Pacific (Seoul)
* Asia Pacific (Singapore)
* EU (Frankfurt)
* EU (Ireland)
* EU (London)
* EU (Paris)
* EU (Stockholm)
* Canada (Central)
* South America (São Paulo)
* AWS GovCloud (US-West)

**More about SQS**

AWS Step Functions is a web service that enables you to coordinate the components of distributed applications and microservices using visual workflows. You build applications from individual components that each perform a discrete function, or *task*, allowing you to scale and change applications quickly. Step Functions provides a reliable way to coordinate components and step through the functions of your application.

### About Amazon Web Services

Amazon Web Services (AWS) is a collection of digital infrastructure services that developers can leverage when developing their applications. The services include computing, storage, database, and application synchronization (messaging and queuing). AWS uses a pay-as-you-go service model: you are charged only for the services that you—or your applications—use. For new AWS users, a free usage tier is available.

* Prerequisites
* Setting Up Step Functions Local

## Create an AWS Account

**Note**

If you previously signed in to the AWS Management Console using AWS account root user credentials, choose **Sign in to a different account**. If you previously signed in to the console using IAM credentials, choose **Sign-in using root account credentials**. Then choose **Create a new AWS account**.

## Create an IAM User

**To create an IAM user for yourself and add the user to an Administrators group**

1. Use your AWS account email address and password to sign in as the AWS account root user to the IAM console at https://console.aws.amazon.com/iam/.

**Note**

We strongly recommend that you adhere to the best practice of using the **Administrator** IAM user below and securely lock away the root user credentials. In the navigation pane of the console, choose **Users**, and then choose **Add user**.

1. For **User name**, type **Administrator**.
2. Select the check box next to **AWS Management Console access**, select **Custom password**, and then type the new user's password in the text box. You can optionally select **Require password reset** to force the user to create a new password the next time the user signs in.
3. Choose **Next: Permissions**.
4. On the **Set permissions** page, choose **Add user to group**.
5. Choose **Create group**.
6. In the **Create group** dialog box, for **Group name** type **Administrators**.
7. For **Filter policies**, select the check box for **AWS managed - job function**.
8. In the policy list, select the check box for **AdministratorAccess**. Then choose **Create group**.
9. Back in the list of groups, select the check box for your new group. Choose **Refresh** if necessary to see the group in the list.
10. Choose **Next: Tags** to add metadata to the user by attaching tags as key-value pairs.
11. Choose **Next: Review** to see the list of group memberships to be added to the new user. When you are ready to proceed, choose **Create user**.

## Step 3: Get Your Access Key ID and Secret Access Key

To use Step Functions actions (for example, using Java or through the AWS Command Line Interface), you need an access key ID and a secret access key.

**Note**

The access key ID and secret access key are specific to AWS Identity and Access Management. Don't confuse them with credentials for other AWS services, such as Amazon EC2 key pairs.

**To get the access key ID and secret access key for an IAM user**

Access keys consist of an access key ID and secret access key, which are used to sign programmatic requests that you make to AWS. If you don't have access keys, you can create them from the AWS Management Console. We recommend that you use IAM access keys instead of AWS account root user access keys. IAM lets you securely control access to AWS services and resources in your AWS account.

The only time that you can view or download the secret access keys is when you create the keys. You cannot recover them later. However, you can create new access keys at any time. You must also have permissions to perform the required IAM actions

1. Open the IAM console.
2. In the navigation pane of the console, choose **Users**.
3. Choose your IAM user name (not the check box).
4. Choose the **Security credentials** tab and then choose **Create access key**.
5. To see the new access key, choose **Show**. Your credentials will look something like this:
   * Access key ID: AKIAIOSFODNN7EXAMPLE
   * Secret access key: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
6. To download the key pair, choose **Download .csv file**. Store the keys in a secure location.

Keep the keys confidential in order to protect your AWS account, and never email them. Do not share them outside your organization, even if an inquiry appears to come from AWS or Amazon.com. No one who legitimately represents Amazon will ever ask you for your secret key.

# Setting Up Step Functions Local

The downloadable version of AWS Step Functions is provided as an executable .jar file, and as a Docker image. The Java application runs on Windows, Linux, macOS X, and other platforms that support Java. In addition to Java, you need to install the AWS Command Line Interface.

**Warning**

The downloadable version of AWS Step Functions is only intended to be used for testing and shouldn't be used to process sensitive information.

Follow these steps to set up and run Step Functions on your computer:

1. Download Step Functions using the following link:

|  |  |
| --- | --- |
| **Download Links** | **Checksum** |
| [.tar.gz](https://s3.amazonaws.com/stepfunctionslocal/StepFunctionsLocal.tar.gz) | [.zip](https://s3.amazonaws.com/stepfunctionslocal/StepFunctionsLocal.zip) | [tar.gz.md5](https://s3.amazonaws.com/stepfunctionslocal/StepFunctionsLocal.tar.gz.md5) | [zip.md5](https://s3.amazonaws.com/stepfunctionslocal/StepFunctionsLocal.zip.md5) |

1. Extract the zip file.
2. Test the download and view version information.
3. $ java -jar StepFunctionsLocal.jar -v
4. Step Function Local
5. Version: 1.0.0

Build: 2019-01-21

1. (Optional) View a listing of available commands:

python abcd.py

$ java -jar StepFunctionsLocal.jar -h

1. To start Step Functions on your computer, open a command prompt window, navigate to the directory where you extracted StepFunctionsLocal.jar and type the following command:

java -jar StepFunctionsLocal.jar

1. To access Step Functions running locally, use the --endpoint-url parameter. For example, using the AWS Command Line Interface, you would specify Step Functions commands as:

aws stepfunctions --endpoint http://localhost:8083 *command*

**Note**

By default Step Functions Local uses a fake account and credentials, and the region is set to US East (N. Virginia). To use Step Functions Local with AWS Lambda, or other supported services, you must configure your credentials and region.

To configure and run Step Functions Local to work with AWS Lambda, Lambda Local, or other supported services, see the following topics.

## Run a HeloWorld State Machine Locally

Once you have run Step Functions locally with the AWS CLI, you can start a state machine execution.

1. Create a state machine from the AWS CLI by escaping the state machine definition.
2. aws stepfunctions --endpoint http://localhost:8083 create-state-machine --definition "{\
3. \"Comment\": \"A Hello World example of the Amazon States Language using a Pass state\",\
4. \"StartAt\": \"HelloWorld\",\
5. \"States\": {\
6. \"HelloWorld\": {\
7. \"Type\": \"Pass\",\
8. \"End\": true\
9. }\

}}" --name "HelloWorld" --role-arn "arn:aws:iam::012345678901:role/DummyRole"

**Note**

The role-arn is not used for Step Functions Local, but you must have it included with the proper syntax. You can use the ARN from the above example.

If you successfully create the state machine, Step Functions will respond with the creation date and the state machine ARN:

{

"creationDate": 1548454198.202,

"stateMachineArn": "arn:aws:states:us-east-1:123456789012:stateMachine:HelloWorld"

}

1. Start an execution using the ARN of the state machine you created.
2. aws stepfunctions --endpoint http://localhost:8083 start-execution --state-machine-arn arn:aws:states:us-east-1:123456789012:stateMachine:HelloWorld

## Step Functions Local with Lambda Local

You can use the local version of Step Functions along with a local version of AWS Lambda. To configure this, you must install and configure the AWS Serverless Application Model (AWS SAM).

For information on configuring and running AWS SAM, see:

* [Set Up AWS SAM](https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/serverless-quick-start.html)
* [Start Lambda Local](https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/sam-cli-command-reference-sam-local-start-lambda.html)

Once Lambda is running on your local system, you can start Step Functions Local. From the directory where you extracted your Step Functions local jar files, start Step Functions Local, configuring the local Lambda endpoint:

java -jar StepFunctionsLocal.jar --lambda-endpoint http://127.0.0.1:3001 *command*

# Getting Started

This tutorial introduces you to the basics of working with AWS Step Functions. You'll create a simple, independently running state machine using a Pass state. The Pass state represents a no-op (an instruction with no operation).

**Topics**

* [Step 1: Creating a State Machine](https://docs.aws.amazon.com/step-functions/latest/dg/getting-started.html#create-state-machine-step-1)
* [Step 2: Starting a New Execution](https://docs.aws.amazon.com/step-functions/latest/dg/getting-started.html#create-state-machine-step-2)
* [Step 3: (Optional) Update a State Machine](https://docs.aws.amazon.com/step-functions/latest/dg/getting-started.html#update-state-machine-step-3)
* [Next Steps](https://docs.aws.amazon.com/step-functions/latest/dg/getting-started.html#next-steps)

## Step 1: Creating a State Machine

Step Functions offers various predefined state machines as templates. Create your first state machine using the **Hello World** template.

* To create the state machine
* Sign in to the [Step Functions console](https://console.aws.amazon.com/states/home), and then choose **Get Started**.
* On the **Define state machine** page, review the **State machine definition** and the visual workflow.
* 
                Hello world.
              
* Step Functions fills in the name of the state machine automatically. It also populates the **Code** pane with the Amazon States Language description of the state machine.
* {
* "Comment": "A Hello World example of the Amazon States Language using a Pass state",
* "StartAt": "HelloWorld",
* "States": {
* "HelloWorld": {
* "Type": "Pass",
* "Result": "Hello World!",
* "End": true
* }
* }
* }
* This JSON text defines a Pass state named HelloWorld. For more information, see [State Machine Structure](https://docs.aws.amazon.com/step-functions/latest/dg/amazon-states-language-state-machine-structure.html).
* Use the graph in the **Visual Workflow** pane to check that your Amazon States Language code describes your state machine correctly.
* 
                Visual Workflow.
              
* If you don't see the graph, choose ![
         refresh
      ]() in the **Visual Workflow** pane.
* Choose **Next**.
* Create or enter an IAM role.
* To create a new IAM role for Step Functions, select **Create an IAM role for me**, and enter a **Name** for your role.
* If you have [previously created an IAM role](https://docs.aws.amazon.com/step-functions/latest/dg/procedure-create-iam-role.html) with the correct permissions for your state machine, select **Choose an existing IAM role**. Select a role from the drop-down, or provide an ARN for that role.
* **Note**
* If you delete the IAM role that Step Functions creates, Step Functions can't recreate it later. Similarly, if you modify the role (for example, by removing Step Functions from the principals in the IAM policy), Step Functions can't restore its original settings later.
* Select **Create state machine**.
* Step 2: Starting a New Execution
* After you create your state machine, you can start an execution.
* To start a new execution
* On the ***Helloworld*** page, choose **New execution**, or **Start execution** if you have started an execution before.
* The **New execution** window is displayed.
* (Optional) To help identify your execution, you can specify an ID for it in the **Enter an execution name** box. If you don't enter an ID, Step Functions generates a unique ID automatically.
* **Note**
* Step Functions allows you to create state machine, execution, and activity names that contain non-ASCII characters. These non-ASCII names don't work with Amazon CloudWatch. To ensure that you can track CloudWatch metrics, choose a name that uses only ASCII characters.
* Choose **Start execution**.
* A new execution of your state machine starts, and a new page showing your running execution is displayed.
* (Optional) In the **Execution Details** section, choose the **Info** tab to view the **Execution Status** and the **Started** and **Closed** timestamps.
* To view the results of your execution, choose the **Output** tab.
* 
                State machine results.
              
* Step 3: (Optional) Update a State Machine
* You can update your state machine for future executions.
* **Note**
* State machine updates in Step Functions are eventually consistent. All executions within a few seconds will use the updated definition and roleArn. Executions started immediately after updating a state machine may use the previous state machine definition and roleArn.
* To update a state machine
* On the ***Helloworld*** page, choose **Edit**.
* The **Edit** page is displayed.
* In the **Code** pane, edit the Amazon States Language description of the state machine. Update the Result to read Hello World has been updated!
* {
* "Comment": "A Hello World example of the Amazon States Language using a Pass state",
* "StartAt": "HelloWorld",
* "States": {
* "HelloWorld": {
* "Type": "Pass",
* "Result": "Hello World *has been updated*!",
* "End": true
* }
* }
* }
* (Optional) Select a new IAM role from the **IAM role for executions** list.
* **Note**
* You can also select **Create new role** to create a new IAM role. For more information,Choose **Save** and then **Start execution**.
* On the **New execution** page choose **Start Execution**.
* To view the results of your execution, select the **HelloWorld** state in the **Visual workflow** and expand the **Output**section under **Step details**.
* 
                Execution output
              
* **Note**
* The output text matches your newly updated state machine.
* Manage a Batch Job (Batch, SNS)
* This sample project demonstrates how to submit a AWS Batch job, and then send an Amazon SNS notification based on whether that job succeeds or fails. Deploying this sample project will create a Step Functions state machine, an AWS Batch job, and an Amazon SNS topic. In this project, Step Functions uses a state machine to call the AWS Batch job synchronously. It then waits for the job to succeed or fail, and it sends an Amazon SNS topic with a message about whether the job succeeded or failed.
* To create the **Manage a Batch Job** state machine and provision all resources:

1. Log in to the [Step Functions console](https://console.aws.amazon.com/states/home?region=us-east-1#/), and choose **Create a state machine**.
2. Select **Sample Projects** and choose **Manage a Batch Job**.

The state machine **Code** and **Visual Workflow** are displayed.


                    Manage AWS Batch workflow.
                

1. Select **Next**.

The **Deploy resources** page is displayed, listing the resources that will be created. For this sample project the resources include:

* + An AWS Batch job
  + An Amazon SNS topic

1. Choose **Deploy Resources**.

**Note**

It can take up to 10 minutes as these resources and related IAM permissions are created. While the **Deploy resources** page displays, you can open the **Stack ID** link to see which resources are being provisioned.

## To start a new execution

1. On the **New execution** page, enter an execution name (optional) and choose **Start Execution.**
2. (Optional) To help identify your execution, you can specify an ID for it in the **Enter an execution name** box. If you don't enter an ID, Step Functions generates a unique ID automatically.

**Note**

Step Functions allows you to create state machine, execution, and activity names that contain non-ASCII characters. These non-ASCII names don't work with Amazon CloudWatch. To ensure that you can track CloudWatch metrics, choose a name that uses only ASCII characters.

1. Optionally, you can go to the newly-created state machine on the Step Functions **Dashboard**, select **New execution**.
2. Once an execution is complete, you can select states on the **Visual workflow** and browse the **Input** and **Output**under **Step details**

## Example State machine code

The state machine in this sample project integrates with AWS Batch and Amazon SNS by passing parameters directly to those resources. Browse through this example state machine to see how Step Functions controls AWS Batch; and Amazon SNS by connecting to the ARN in the Resource field, and by passing Parameters to the service API.

{

"Comment": "An example of the Amazon States Language for notification on an AWS Batch job completion",

"StartAt": "Submit Batch Job",

"TimeoutSeconds": 3600,

"States": {

"Submit Batch Job": {

"Type": "Task",

"Resource": "arn:aws:states:::batch:submitJob.sync",

"Parameters": {

"JobName": "BatchJobNotification",

"JobQueue": "arn:aws:batch:us-east-1:123456789012:job-queue/BatchJobQueue-7049d367474b4dd",

"JobDefinition": "arn:aws:batch:us-east-1:123456789012:job-definition/BatchJobDefinition-74d55ec34c4643c:1"

},

"Next": "Notify Success",

"Catch": [

{

"ErrorEquals": [ "States.ALL" ],

"Next": "Notify Failure"

}

]

},

"Notify Success": {

"Type": "Task",

"Resource": "arn:aws:states:::sns:publish",

"Parameters": {

"Message": "Batch job submitted through Step Functions succeeded",

"TopicArn": "arn:aws:sns:us-east-1:123456789012:batchjobnotificatiointemplate-SNSTopic-1J757CVBQ2KHM"

},

"End": true

},

"Notify Failure": {

"Type": "Task",

"Resource": "arn:aws:states:::sns:publish",

"Parameters": {

"Message": "Batch job submitted through Step Functions failed",

"TopicArn": "arn:aws:sns:us-east-1:123456789012:batchjobnotificatiointemplate-SNSTopic-1J757CVBQ2KHM"

},

"End": true

}

}

}

## IAM Example

This example IAM policy generated by the sample project includes the least privilege necessary to execute the state machine and related resources. It is a best practice to include only those permissions necessary in your IAM policies

{

"Version": "2012-10-17",

"Statement": [

{

"Action": [

"sns:Publish"

],

"Resource": [

"arn:aws:sns:ap-northeast-1:123456789012:ManageBatchJob-SNSTopic-JHLYYG7AZPZI"

],

"Effect": "Allow"

},

{

"Action": [

"batch:SubmitJob",

"batch:DescribeJobs",

"batch:TerminateJob"

],

"Resource": "\*",

"Effect": "Allow"

},

{

"Action": [

"events:PutTargets",

"events:PutRule",

"events:DescribeRule"

],

"Resource": [

"arn:aws:events:ap-northeast-1:123456789012:rule/StepFunctionsGetEventsForBatchJobsRule"

],

"Effect": "Allow"

}

]

}