AWS DeepComposer

Developer Guide

AWS DeepComposer: Developer Guide

Table of Contents

What is AWS DeepComposer?

AWS DeepComposer provides a creative, hands-on experience for learning generative AI and machine learning. With generative AI, one of the biggest recent advancements in artificial intelligence, you can create a new dataset based on a training dataset. With AWS DeepComposer, you can experiment with different generative AI architectures and models in a musical setting by creating and transforming musical inputs and accompaniments.

Regardless of your experience with machine learning (ML) or music, you can use AWS DeepComposer to develop a working knowledge of generative AI. AWS DeepComposer includes learning capsules, sample code, and training data to help you understand and use generative AI models.

To get started with AWS DeepComposer, start the AWS DeepComposer Music studio, choose one of the sample melodies, and choose a pretrained model. After you generate a composition, you can change the instruments, download your new composition, and share it with friends on SoundCloud.

To flex your creativity, record a custom melody using either the console-based keyboard or the AWS DeepComposer keyboard. To dive deeper, start training custom models using training data provided by AWS. Want more? Learn how to create your own generative adversarial networks (GANs) by using the examples in Jupyter notebooks for SageMaker.

> **Important - Browser requirements**
> AWS DeepComposer fully supports the Chrome browser. Other browsers offer limited support for the AWS DeepComposer console and hardware. For more information about browser compatibility, see *Browser support* (p. 52).

Topics

AWS DeepComposer concepts and terminology

AWS DeepComposer builds on the following concepts and uses the following terminology.

autoregressive convolutional neural network (AR-CNN)

A generative AI technique that edits your input melody. This technique uses a U-Net architecture and was trained on a collection of chorales by Johann Sebastian Bach. The AR-CNN technique works by detecting notes that sound missing or out place based on the training dataset. The identified notes are then replaced with notes that the model thinks would fit the distribution of notes learned during training.

AutoregressiveCNN Bach

A pretrained autoregressive convolutional neural network (AR-CNN) model that is available in the AWS DeepComposer music studio. This model has been trained on a dataset containing only Bach chorales. It uses the U-net architecture.

AWS DeepComposer keyboard

Also known as the AWS DeepComposer physical keyboard or hardware keyboard. You can connect, or link, the keyboard to a computer that has access to the AWS DeepComposer console. You can use the linked keyboard to play and record short melodies that are fewer than eight bars. Then use a recorded melody with a supported AWS DeepComposer generative AI technique.

compositions

Sequences of notes, melodies, harmonies, and rhythms that make up a musical work.

In AWS DeepComposer, a composition also represents a saved piece of music. Each time you use a generative technique to perform inference, your output is automatically saved as a composition.

convolutional neural network (CNN)

A type of neural network commonly used for image-recognition and video-recognition tasks. A CNN filters and summarizes the data during training to learn patterns. This mathematical process is called *convolution*.

decoder

A process that takes an input vector, commonly a feature matrix, and transforms the vector into an output. A decoder can be used with an encoder. AWS DeepComposer uses an encoder-decoder network for some generative AI tasks.

discriminator

A classifier model, one part of a generative adversarial network (GAN). The discriminator classifies data as real or as generated. In AWS DeepComposer, this model tries to determine if a generated piano roll looks like a real image or an artificially created image. The discriminator is trained on real data.

drum in pattern

A ratio of the total number of notes in a drum track to a predetermined beat popular in 4/4 time.

empty bar rate

The ratio of empty bars to the total number of bars.

encoder

A process that translates data from one format to another. AWS DeepComposer uses multiple encoders. For example, it uses an encoder to translate an input MIDI file into a piano roll image. Often, an encoder is used with a decoder.

epoch

One complete pass through the training dataset by the neural network. For example, if you have 10,000 music tracks in the training dataset, one epoch represents one pass through all 10,000 tracks. The number of epochs required for a model to converge varies based on the training data. An iterative algorithm converges when the loss function stabilizes.

generative adversarial network (GAN)

Two neural networks that consist of a *generator* and a *discriminator*. In AWS DeepComposer, the generator learns to compose music that sounds as realistic as possible with feedback from the discriminator. The discriminator treats the generator's output as sounding as unrealistic as possible while holding the input training sample as the ground truth. Training proceeds, with the generator searching for its network weights by minimizing the chances that its generations differ from the training samples. The discriminator searches for the weights of both networks by minimizing the chances it misjudges the training samples as realistic compositions and the generator's outputs as unrealistic compositions. The efforts made by both the generator and the discriminator push the generator's network weights in opposite directions. This is the essence of the adversarial role that the discriminator takes against the generator. Learning proceeds until equilibrium is reached, where the generator improves its output to such a degree that the discriminator can no longer distinguish between the generated composition and training samples.

hyperparameters

Algorithm-dependent variables that you can control. You can tune hyperparameters to find the best fit for a specific problem that you are trying to model.

inference

Predictions generated by a trained model.

in scale ratio

The ratio of the average number of notes in a bar of music that are in the key of C, to the total number of notes in a bar of music.

learning rate

A hyperparameter used for training neural networks. The learning rate controls how much the weights and biases are updated during training.

loss function

Evaluates how effective an algorithm is at modeling the data. For example, if a model consistently predicts values that are very different from the actual values, it returns a large loss. Depending on the training algorithm, more than one loss function might be used.

model

The final output created while training with a machine learning algorithm. The algorithm used to train the model finds patterns in the training data uses those patterns to map the input data attributes to the target (the answer that you want to predict). The algorithm outputs a machine learning model that captures these patterns.

MuseGAN

A generative adversarial network (GAN) architecture built specifically to generate music. Like other GANs, MuseGAN is made of both a discriminator and a generator that use a CNN. The MuseGAN architecture is available in AWS DeepComposer. To learn more about the MuseGAN architecture, see Creating the MuseGAN architecture.

Music studio

A component of the AWS DeepComposer console that provides access to a console-based keyboard for playing, recording, and composing music. With Music studio, you can try out AWS DeepComposer before purchasing an AWS DeepComposer physical keyboard. You don't need to train a model to use Music studio.

neural network

Also known as an artificial neural network. A collection of connected units or nodes that are used to build an information model based on biological systems. Each node is called an artificial neuron. An artificial neuron mimics a biological neuron in that it receives an input (stimulus), becomes activated if the input signal is strong enough (activation), and produces an output predicated on the input and activation. Neural networks are widely used in machine learning because an artificial neural network can serve as a general-purpose approximation to any function.

pitch

The frequency of a sound, often judged to be high or low. In the AWS DeepComposer Music studio, you can adjust the pitch of a sample, imported, or custom recorded input melody.

pitches used

A metric that captures the average number of notes in each bar.

polyphonic rate

The ratio of the number of time steps where the number of pitches being played is greater than the **threshold** number of allowable time steps.

Rhythm assist

An AWS DeepComposer function that automatically corrects the timing of musical notes in your input. Use Rhythm assist when your melody contains the right notes but isn't consistently in time with the beat. Rhythm assist is available in Music studio.

tempo

How fast music is played. Music typically follows a certain beat or meter, which drives the rhythm of the notes played. The speed of this beat is measured in beats per minute. A higher number of beats per minute corresponds to a faster tempo, or playback speed.

Transformers

A generative AI technique that uses attention to understand the relationship between notes in a musical composition. To train the model, the musical data are converted into tokens. These tokens are used to represent a musical event in a given piece of music. During inference, the Transformers technique will extend your input track by up 30 seconds.

TransformerXLClassical

A model based on the Transformer architecture. Compared with the traditional Transformer architecture, this model can better understand long-term dependencies and has decreased latency times.

U-Net

A generative adversarial network (GAN) architecture built originally for image recognition tasks. The U-Net is a CNN. It's named for its U-like shape, in which the layers on the left side can pass information to the layers on the right side without passing through the entire neural network.

update ratio

A hyperparameter that controls the number of model weight updates to the discriminator per update to the generator. A lower update ratio makes a stronger discriminator that can provide more accurate and useful information to the generator. The lower update ratio, however, increases training time.

virtual keyboard

Also known as the AWS DeepComposer console-based keyboard. The virtual keyboard is available in the AWS DeepComposer Music studio. The virtual keyboard runs on any computer or mobile device that is connected to the AWS Cloud. You can use the virtual keyboard to play and record a short melody. You can then use the recorded melody with a supported AWS DeepComposer generative AI technique to create new musical compositions.

AWS DeepComposer keyboard

The AWS DeepComposer keyboard is a music keyboard that you can use to get hands-on experience learning generative AI and machine learning (ML). You can connect it to any computer that has access to the AWS DeepComposer console, play and record a short melody, and feed the melody to a supported generative AI architecture supported by AWS DeepComposer.

Setting up the AWS DeepComposer keyboard

To get started using your AWS DeepComposer keyboard, link your device inside the AWS DeepComposer console. To do so, follow the steps outlined below.

Connecting the AWS DeepComposer keyboard

1. Open the link your keyboard section of the AWS DeepComposer console.
2. Use the included USB cable to connect the keyboard to your computer.
3. On the back of the keyboard, locate the 8- or 16-digit alphanumeric serial number (S/N or DSN).
4. In **step 2** in the AWS DeepComposer console, enter the alphanumeric serial number.

Important - Browser requirements
AWS DeepComposer fully supports the Chrome browser. Other browsers offer limited support for the AWS DeepComposer console and hardware. For more information about browser compatibility, see *Browser support* (p. 52).

AWS DeepComposer keyboard features

The AWS DeepComposer keyboard is 32-key, 2 octave keyboard. It has several built-in features that are designed to increase the usability of the AWS DeepComposer keyboard.

Arpeggiator (arp)

To create an arpeggio, use this button to step through a group of notes in a particular sequence to create an arpeggio.

Auto-chord (chord)

Use this button to generate a chord by playing a single note. When you play a note, the auto-chord feature generates a simple triad chord using the note provided as the root note.

Modulation

Use this wheel to modulate the synthetic audio signal to create a vibrato effect.

Octave adjust

The default starting position of the AWS DeepComposer keyboard is the C4 and C3 octaves. The octave down button adjusts the keyboard down one or more octaves. The octave up button adjusts the keyboard up one or more octaves.

Pitch

Use this wheel to slightly bend (change) the pitch of a note up or down.

Playback

Use this button to playback a melody that you just recoreded.

Record

Use this button to begin recording a melody on your keyboard. To stop recording, tap the button again.

Sustain switch interface

You can plug any universal MIDI-compliant sustain pedal into your keyboard through this port.

Tap tempo

Tap this button to set the arpeggio tempo manually.

USB cable port

Use this port to connect the physical keyboard to the AWS DeepComposer console (Music studio).

Volume

Use this slider enables to adjust the volume on your keyboard.

How AWS DeepComposer works

At a fundamental level, music composition corresponds to sequences of notes of intricate tempo and dynamics. Musical compositions are categorized into genres based on discernible differences in the distribution different musical elements. To compose music for specific genre you have learn that genre specific distribution. At its core that is what the different generative AI techniques in AWS DeepComposer allow you to do.

This process involves figuring out the appropriate patterns and how to arrange them together in cohesive way. Aspiring composers typically have to spend several years training to learn these patterns. When composers graduate, they have developed an acute appreciation of the distributions of different musical elements in the genres they studied. Their understanding might be built on non-quantitative terms but they have built a quantitative distribution into their brains. They have developed a a natural neural network model of music composition and mastered ways to apply the model with great proficiency.

In machine learning, it's analogous to teaching a machine to compose music of a given genre. The machine learns the compositional features or patterns from a known musical collection to develop a practical understanding of the distribution of musical elements. The knowledge is built into an artificial neural network and represented by a set of optimal network weights of a chosen architecture.

Getting started with AWS DeepComposer

When you use the AWS DeepComposer Music studio experiences, your creativity and and generative AI can come together. To start using AWS DeepComposer, you need a *trained model* and an *input track*. You can use AWS DeepComposer to train *custom models*, or you can use *sample models*. For your input track, you can use a sample input track, record a custom input track, or import a track.

To create a composition, AWS DeepComposer performs inference (p. 3) with a trained model and an input track. After you create your first few compositions, you can start modifying the hyperparameters (p. 2) available in the AWS DeepComposer Music studio to learn more about how generative AI works.

Right now, you can experiment with creating music in the new remixed music studio experience or continue using the classic music studio experience.

In the following topics, you can learn how to use both classic and remixed music studio experiences to perform inference using different generative AI techniques.

You can also train a custom model in the AWS DeepComposer console using a supported generative AI technique. Then, you can use it in the AWS DeepComposer Music studio to generate compositions.

AWS DeepComposer supports training custom models using the generative adversarial networks (GANs) (p. 2) technique. You can train a model that uses either the MuseGAN (p. 3) or the U-Net (p. 4) algorithm.

If you are new to machine learning or generative AI, see learning the basics of generative AI to get started. If you want a deeper dive into the different generative AI techniques supported by AWS DeepComposer, the learning capsules provide an introduction.

> **Note**
> To use the AWS DeepComposer console and other AWS services, you need an AWS account. If you don't have an account, see aws.amazon.com and choose **Create an AWS Account**. For detailed instructions, see How do I create and activate a new AWS account?.
> As a best practice, you should also create an AWS Identity and Access Management (IAM) user with administrator permissions and use that for all work that does not require root credentials. Create a password for console access, and access keys to use command line tools. For more information, see Creating your first IAM admin user and group in the *IAM User Guide*.

After you've created your AWS account, sign in to the AWS DeepComposer console with your AWS account or user credentials. The following exercises assume you've signed in to the AWS DeepComposer console.

> **Important - Browser requirements**
> AWS DeepComposer fully supports the Chrome browser. Other browsers offer limited support for the AWS DeepComposer console and hardware. For more information about browser compatibility, see *Browser support* (p. 52).

Topics

- Evaluating a model (p. 17)

Creating a composition with a trained model in an AWS DeepComposer Music studio experience

To generate, create, and edit compositions with AWS DeepComposer, you use the AWS DeepComposer Music studio. To get started, you need a trained model and an input track.

For the model, you can use either a sample or a custom model. This topic covers how to use a *sample model*. Sample models are available in both AWS DeepComposer Music studio experiences.

For the input track, you can use a sample track, record a custom track, or import a track.

Each AWS DeepComposer Music studio experience supports three different generative AI techniques, generative adversarial networks (GANs) (p. 2), autoregressive convolutional neural network (AR-CNN) (p. 1), and Transformers (p. 4). You can use the GANs technique to create accompaniment tracks. You can use the AR-CNN technique to modify notes in your input track. You can use the Transformers technique to extend your input track by up to 30 seconds.

To learn more about how to use the different generative techniques together, see the Generative techniques in AWS DeepComposer (p. 22) topic. The topic covers how to collaborate interactively with the generative techniques in AWS DeepComposer. You can also learn how to best use the available techniques together.

In each of the following topics, you can find directions for using either AWS DeepComposer Music studio experience.

Topics

- Using a sample input melody to create a composition (p. 10)
- Recording a custom melody to create a composition (p. 11)
- Using an imported track to create a composition (p. 14)

Using a sample input melody to create a composition

This topic shows you how to use a sample input melody and a sample model to create your first composition in under 5 minutes. Using a sample input track is the quickest way to get started with AWS DeepComposer. The procedures below will work with any sample or previously trained custom model. following procedures.

Generating a composition using the classic music studio experience

To generate a composition with a sample input melody

1. Open the AWS DeepComposer console.
2. In the navigation pane, choose **Music studio**.
3. To open the **Input melody** section, choose the right arrow (▸).
4. Choose the down arrow (▾) to open the **Sample track** menu.
5. Choose **Ode to Joy**.
6. Under **Model parameters**, for **Generative AI technique**, choose **AR- CNN**.
7. Choose **Enhance input melody**.

After inference is complete you can do the following:

- Change the **Advanced parameters**, choose **Enhance input melody** , and then choose **Play** to hear how your track has changed.
- Choose **Edit melody** to modify and change the notes that were added during inference.

Important
Each time you perform one of the preceding modifications, your composition is automatically save as a *new composition*.

8. To listen to your new composition, choose **Play** (▶).

Generating a composition using the remixed music studio experience

To generate a composition with sample input track

1. Open the AWS DeepComposer console.
2. In the navigation pane, choose **Try the remixed music studio**.
3. To get started, choose **Start composing**.
4. On the **Input track** page, under **Choose an input track**, choose the **down arrow** (▼).
5. Choose **Ode to Joy**.
6. Choose **Continue**.
7. On the **ML technique** page, choose **AR-CNN**.
8. Choose **Continue**.
9. On the **Inference output** page, you can do the following:

   - Change the **AR-CNN parameters**, choose **Enhance again**, and then choose **Play** to hear how your track has changed.
   - Choose **Edit melody** to modify and change the notes that were added during inference.

   Important
   Each time you perform one of the preceding modifications, your composition is automatically save as a *new composition*.
10. Choose **Continue** to finish creating your composition in the remixed music studio.

The AWS DeepComposer Music studio experiences are a great place to experiment and play. You can start by choosing a different generative AI technique to use with your sample input track or use the different techniques collaboratively to create your next musical masterpiece.

To learn more about how you can use the different generative techniques together, see the Generative techniques in AWS DeepComposer (p. 22) topic.

Recording a custom melody to create a composition

To record a custom input melody, you can use the AWS DeepComposer built-in virtual keyboard (p. 4) or a registered AWS DeepComposer keyboard.

Recording a melody using the classic music studio experience

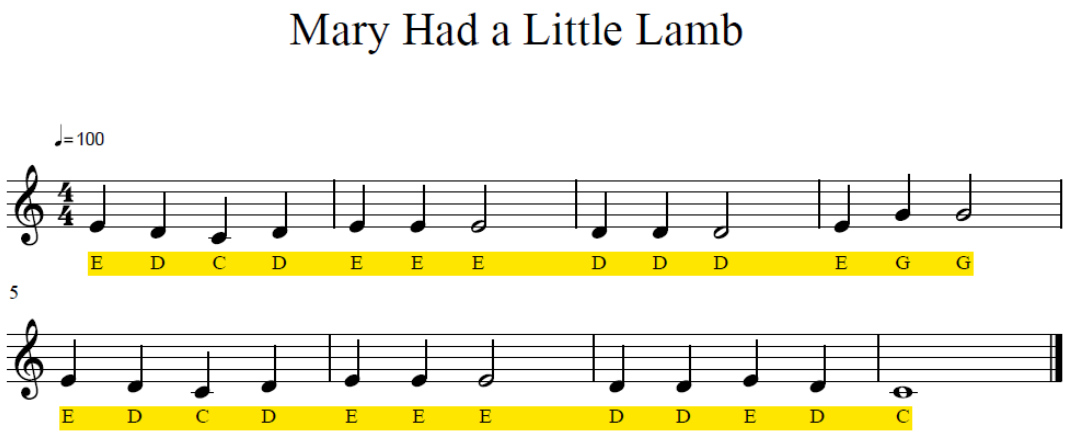
In the classic music studio experience, the **Settings** icon (#) icon is located on the **Music studio** landing page. Choose the icon to activate, or deactivate, the following settings:

- **Metronome** turns on or off a metronome that ticks while you record your melody. After turning on the metronome, use the slider on the music studio console to adjust the speed.
- **Hot keys** enables the use of your computer keyboard for recording your melody.
- **Countdown** provides 5 seconds of lead-in time before recording begins so that you can prepare for recording.

To record a custom melody and use it to create a composition

1. Open the AWS DeepComposer console.
2. In the navigation pane, choose **Music studio**.
3. Choose **Record** (●).

   If you need a little inspiration, try recording this melody.



4. When you've finished, choose **Stop** (■).
5. To listen to your new recording, choose **Play** (▸).
6. (Optional) To fix notes that sound out place or to add notes that you missed while recording, choose **Edit melody**.
7. In the **Model parameters** section, for **Generative AI technique**, choose **AR-CNN**.
8. Choose **Enhance input melody**.

   After inference is complete, you can do the following:

   - Change the **Advanced parameters**, choose **Enhance again**, and then choose **Play** to hear how your track has changed.
   - Choose **Edit melody** to modify and change the notes that were added during inference.

   Important
   Each time you perform one of the preceding modifications, your composition is automatically save as a *new composition*.
9. To listen to your new generative AI musical composition, choose **Play** (▸).

Recording a track using the remixed music studio experience

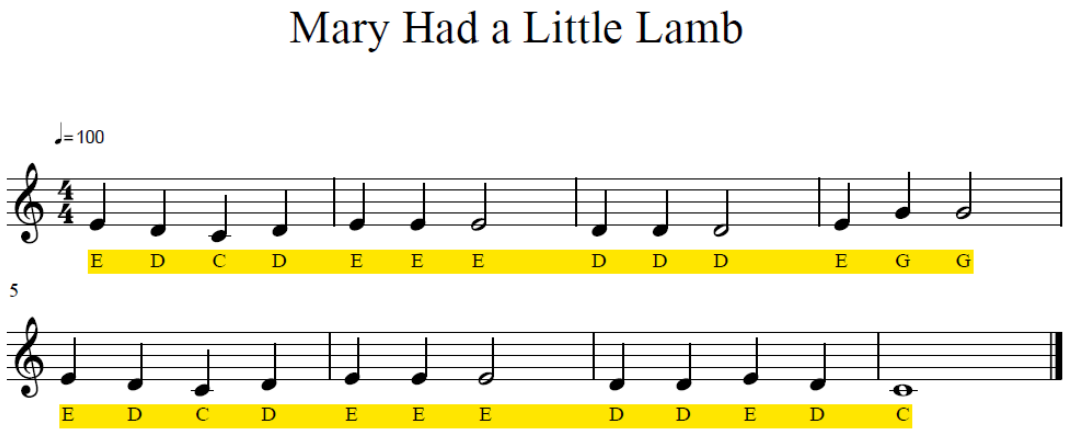
In the remixed music studio, the **Settings** icon (#) is located on the **Input track** page. Choose the icon to activate, or deactivate, the following settings:

- **Metronome** turns on or off a metronome that ticks while you record your melody. After turning on the metronome, use the slider on the music studio console to adjust the speed.
- **Hot keys** enables the use of your computer keyboard for recording your melody.
- **Countdown** provides 5 seconds of lead-in time before recording begins so that you can prepare for recording.

To record a custom melody and use it to create a composition

1. Open the AWS DeepComposer console.
2. In the navigation pane, choose **Try the remixed music studio**.
3. To get started, choose **Start composing**.
4. On the **Input track** page, choose **Record** (●).

   If you need a little inspiration, try recording this melody.



5. When you've finished, choose **Stop** (■).
6. To listen to your new recording, choose **Play** (▸).
7. (Optional) To fix notes that sound out place or to add notes that you missed while recording, choose **Edit melody**.
8. Choose **Continue**.
9. On the **ML technique** page, choose **AR-CNN**.
10. Choose **Continue**. AWS DeepComposer performs inference to create a new composition.
11. On the **Inference output** page, you can do the following:

    - Change the **AR-CNN parameters**, choose **Enhance again**, and choose **Play** to hear how your track has changed.
    - Choose **Edit melody** to modify and change the notes that were added during inference.

    Important
    Each time you perform one of the preceding modifications, your composition is automatically save as a *new composition*.
12. When you have finished, choose **Continue**.

The AWS DeepComposer Music studio experiences allow you to explore and play with different supported generative AI techniques. You can try using a different generative AI techniques with your

custom recording. The different generative AI techniques can be used together. To learn more, see the topic on generative techniques in AWS DeepComposer (p. 22).

Using an imported track to create a composition

With AWS DeepComposer Music studio, you can import your own input track. You can import tracks that are saved as .midi files and are less than 10 KB.

> **Note**
> You certify that you own all necessary rights to any content you import and you acknowledge and agree that you will not use AWS DeepComposer to violate or infringe the intellectual property rights of others.

Importing a track in the classic music studio experience

To import a track and then use it to create a composition

1. Open the AWS DeepComposer console.
2. In the navigation pane, choose **Music studio**.
3. To open the **Input melody** section, choose the right arrow (▸).
4. Change **Source of input melody** to **Imported track**.
5. To import your track, choose **Choose file**.
6. Under **Model parameters**, for **Generative AI technique**, choose **AR-CNN**.
7. Choose **Enhance input melody**.

   After inference is complete, you can do the following:

   - Change the **Advanced parameters**, choose **Enhance input melody** to perform inference again, and then choose **Play** to hear how your track has changed.
   - Choose **Edit melody** to modify and change the notes that were generated during inference.

   > **Important**
   > Each time you perform one of the preceding modifications, your composition is automatically save as a *new composition*.
8. To listen to your new musical composition created with generative AI, choose **Play** (▸).

Importing a track in the remixed music studio experience

To import a track and then use it to create a composition

1. Open the AWS DeepComposer console.
2. In the navigation pane, choose **Try the remixed music studio**.
3. Choose **Start composing**.
4. On the **Input track** page, under **Choose an input track**, choose the **down arrow** (▾).
5. Choose **Import a track**
6. In the **Import a track** dialog box, choose **Choose a file**. Then use the file explorer on your computer to select and upload your file.
7. Choose **Save**.
8. Choose **Continue**.

9. On the **ML technique** page, choose **AR-CNN**.
10. Choose **Continue**. AWS DeepComposer performs inference to create a new composition.
11. On the **Inference output** page, you can do the following:

    - Change the **AR-CNN parameters**, choose **Enhance again** to perform inference again, and then choose **Play** to hear how your track has changed.
    - Choose **Edit melody** to modify and change the notes that were generated during inference.

    Important
    Each time you perform one of the preceding modifications, your composition is automatically save as a *new composition*.
12. When you finished, choose **Continue**.

To learn more, explore the generative techniques in AWS DeepComposer (p. 22) topic. This topic explains how you can work more collaboratively with the available generative techniques.

Training a custom model for the AWS DeepComposer Music studio

You can train a custom model in the AWS DeepComposer console using a supported generative AI technique and algorithm. Then you can use the model in the AWS DeepComposer music studio to generate compositions.

AWS DeepComposer supports training custom models through the generative adversarial network (GAN) (p. 2) architecture. You can train a model using the MuseGAN (p. 3) or U-Net (p. 4) algorithm. Each algorithm supports different training datasets (p. 18). To use the symphony, jazz, pop, or rock genre-based datasets, choose the MuseGAN algorithm. The U-Net algorithm is limited to a training dataset based on music by Johann Sebastian Bach.

After training a custom model, you can perform inference (p. 3) using an input track in the AWS DeepComposer music studio. When you perform inference, you use your custom trained model to generate a new composition.

After training a custom model, you can perform inference (p. 3) using an input track in the AWS DeepComposer Music studio. When you perform inference, you use your custom trained model to generate a new composition.

If you use the default values to train your custom model, training will take approximately 8 hours. To decrease the amount of time, you can decrease the number of training epochs (p. 2). Decreasing the number of training epochs by too much, however, might result in a poorly trained model.

Note
Before proceeding further, make sure you have signed up for an AWS account and signed in to the AWS DeepComposer console (p. 9).

To train a custom model

1. Open the AWS DeepComposer console.
2. In the navigation pane, choose **Models**.
3. On the **Models** page, choose **Create a model**.
4. On the **Train a model** page, under **Generative algorithm**, choose **MuseGAN**.

Both the MuseGAN and U-Net algorithms are based on convolutional neural networks (CNNs). These CNNs generate a high-level image-based representation of music called a piano roll, which is used to generate the output tracks.

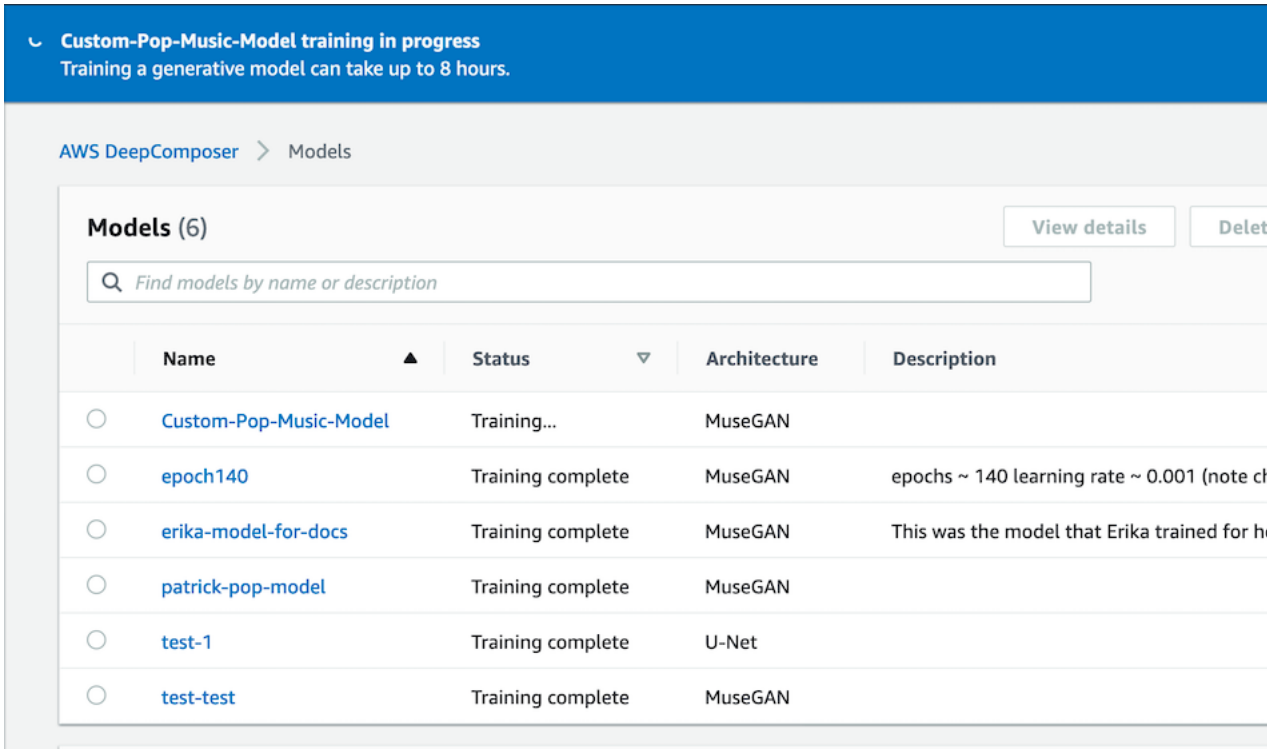
5. For **Training dataset**, choose **jazz**.
6. For **Hyperparameters**, choose the following values.

   Hyperparameters are algorithm-dependent variables that you control. You can tune the hyperparameters to find the best fit for the specific problem that you are trying to model.

Hyperparameter	Value
Epoch (p. 2)	100
Learning rate (p. 3)	0.001
Update ratio (p. 4)	4

7. Under **Model details**, give your model a name and an optional description.
8. Choose **Start training**.
9. To check the status of training, in the navigation pane, choose **Models**. The status of the training appears under **Status**.

   After the model has been trained successfully, the status changes to **Training complete**.



After training has completed, your model automatically appears in either AWS DeepComposer Music studio experience, where you can use it to create compositions.

To learn more about using your custom model, refer to the topic on creating compositions with a trained model in AWS DeepComposer (p. 10).

Evaluating a model

By examining trained models, you can learn what a useful model is and the features that a model should include. Always evaluate a model to understand the predictions that it generates. To evaluate a model, you can examine the changes in the loss function of your model over time. You can also explore the training output per 50th epoch on the model details page. This topic covers what makes an effective model and which hyperparameters are available for different models.

> **Important**
> This topic assumes that you chose the hyperparameters documented in the topic on training a custom MuseGAN model (p. 15). If you chose another model or different hyperparameters, your results will differ from those shown in this topic.

Topics

Key words and phrases

Before examining models, review the following terms and definitions. The full list of available terms is available in the topic on AWS DeepComposer concepts and terminology (p. 1).

discriminator

A classifier model, which is one part of a generative adversarial network (GAN). The discriminator classifies real data from generated data. In the AWS DeepComposer use case, this model tries to determine if a generated piano roll looks like a real piano roll or an artificially created piano roll image. The discriminator is trained on real data.

epoch

One complete pass through the training dataset by the neural network. For example, if you have 10,000 music tracks in the training dataset, one epoch represents one pass through all 10,000 tracks. The number of epochs required for a model to converge varies based on the training data. An iterative algorithm converges when the loss function stabilizes.

generator

One part of a GAN. The generator creates new data by collecting and incorporating feedback from the discriminator. A GAN is made up of two models: a generator and a discriminator. Both have separate loss functions. The goal of the generator is to create the most accurate data possible. The goal of the discriminator is to determine whether that data is accurate.

learning rate

A training hyperparameter that controls how large the step size is at each iteration and update. Step size is how much model weights and parameters change. A small learning rate requires more epochs because smaller changes are made to each update. A larger learning rate results in rapid changes. A learning rate that is too large can cause the model to converge too quickly and pass the optimal point. A learning rate that is too small can cause the training process to get stuck before reaching the optimal point.

loss function

Controls how accurate a prediction is. For example, if an autonomous car misidentifies a pedestrian as street marks, it results in a harmful outcome for both parties. Loss functions help prevent these kinds of mistakes by mitigating the errors and depicting by how much the algorithm has missed the desired target. In this example, the correct loss function would prevent the car's algorithm from mistaking the pedestrian as street marks by ensuring that the prediction was as accurate as possible.

training dataset

A dataset that helps a model learn about the music that you're training. You choose the training dataset based on the two model types provided: MuseGAN or U-Net. If you want to train a model for the symphony, jazz, pop, or rock genres, use a MuseGAN model. Datasets for each of those genres are provided. The dataset for U-Net models is limited to the Bach genre.

update ratio

The number of generator updates compared to discriminator updates in GANs. After the generator is trained, the discriminator must be trained. For example, an update ratio of **5** means that for every five times the discriminator is updated, the generator is updated one time.

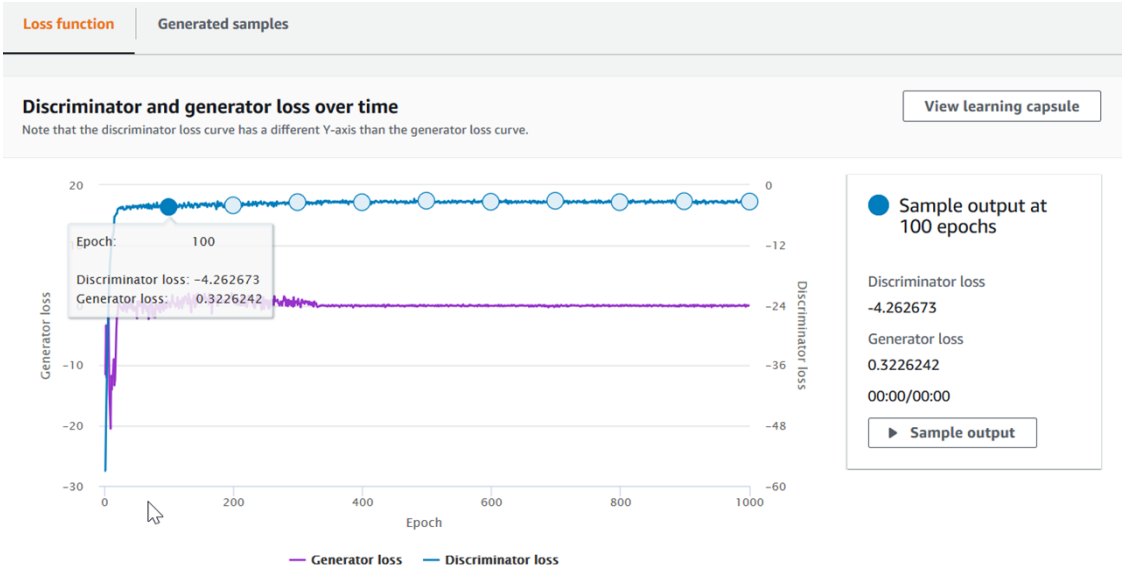
Evaluating the MuseGAN jazz sample model

In this section, you review the loss function and generated sample metrics that are provided in the AWS DeepComposer console for the sample MuseGAN jazz genre model.

> **Note**
> In the AWS DeepComposer console, the generated samples metrics are available only for pre-trained models. These metrics include **Pitches used**, **Pitch classes used**, **Empty bar rate**, **Polyphonic range**, **In scale ratio**, and **Drum in pattern**.

To evaluate the pre-trained MuseGAN jazz genre model

1. Open the AWS DeepComposer console.
2. In the navigation pane, choose **Models**.
3. On the **Models** page, under **Sample model**, choose the **Jazz** pre-trained model.
4. On the training results page for the model, choose **Loss function** to review the loss function graph.



The generator loss and discriminator loss are superimposed onto the same graph, but on different scales. The scale of the generator loss is shown on the left side and the scale of the discriminator loss is shown on the right side.

In this training job, the generator loss plateaus around the 50th epoch, when it stops significantly improving its ability to generate realistic music.

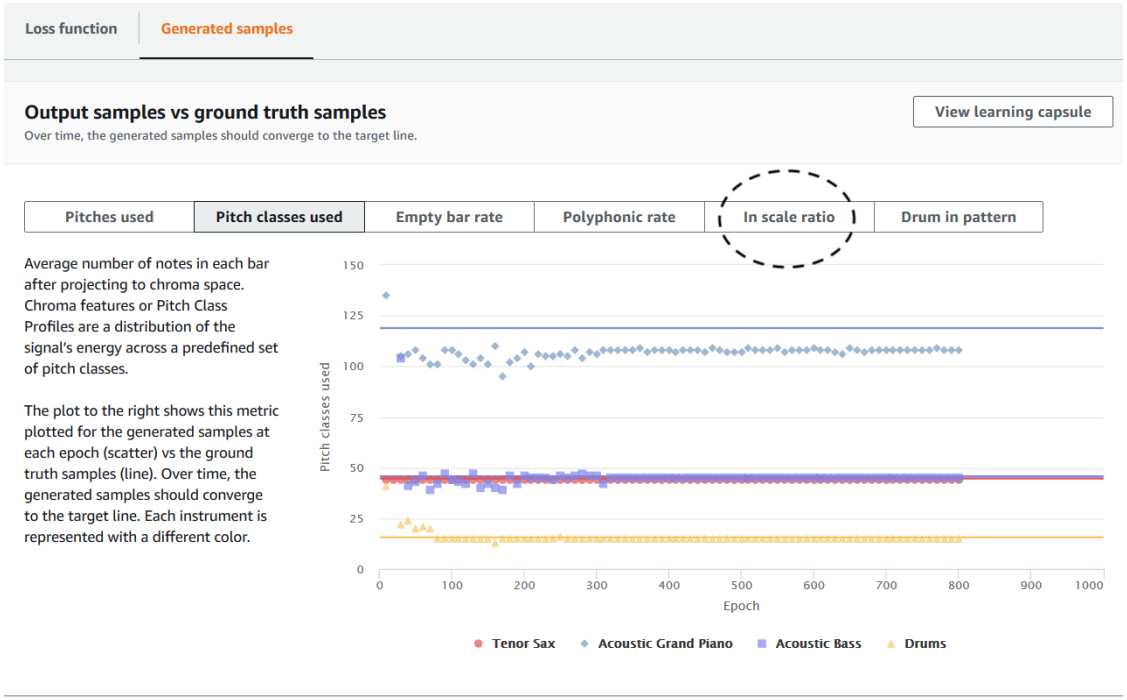
5. Choose **Sample output** to listen to accompaniment tracks that would be generated had inference been performed at that specific epoch.

   For every 50th epoch, you can listen to the accompaniment tracks that could have been generated.

   The discriminator loss shows similar behavior, but is less noisy after it plateaus.
6. Choose the **Generated samples** tab.

   You can choose one of the available metrics: **Pitches used**, **Pitch classes used**, **Empty bar rate**, **Polyphonic range**, **In scale ratio**, **Drum in pattern**.

   To view the definition of a generated sample metric, choose the corresponding tab. For each generated sample, a graph shows output samples compared with ground truth samples. Over time, the generated samples should converge to the target line. The plot in the tab shows the metric plotted for the generated samples at each epoch (scatter) vs. the ground truth samples (line), respectively. The ground truth samples allow image data to be related to real images as opposed to images provided by inference.



Evaluating a custom model

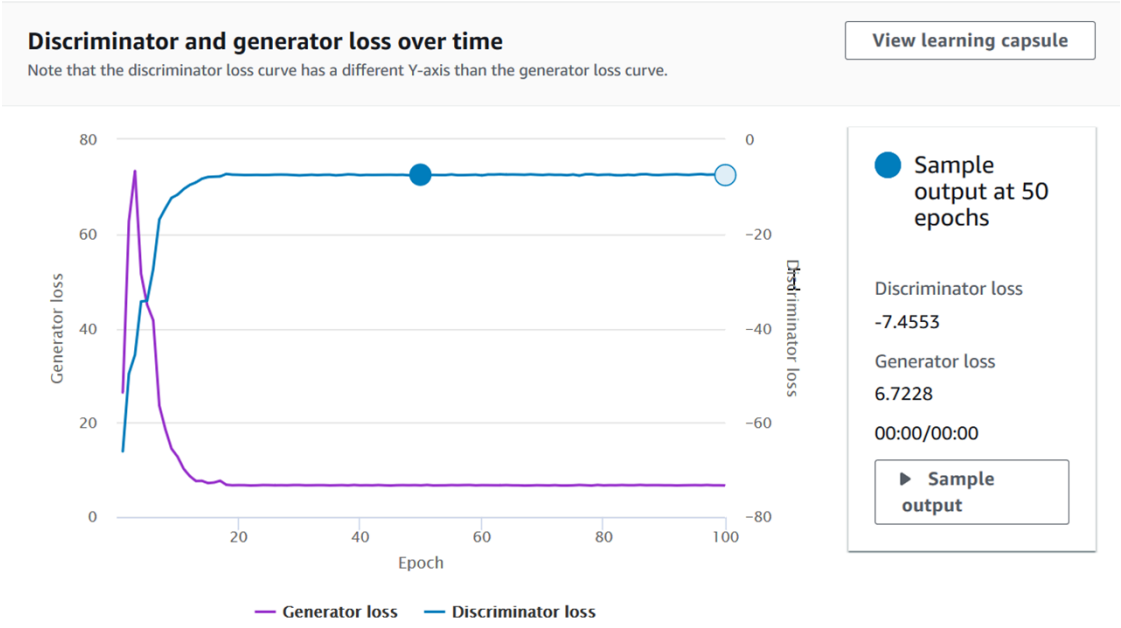
In this topic, you learn how to evaluate the loss function graph on the **Models** page, and what to do when your training is unsuccessful.

> **Important**
> This topic assumes that you have chosen the hyperparameters as documented in the topic on training a custom MuseGAN model (p. 15). If you chose different hyperparameter values, your results will be different.

To evaluate a custom model

1. Open the AWS DeepComposer console.
2. In the navigation pane, choose **Models**.
3. On the **Models** page, choose your custom model.

4. On the model's training results page, under **Discriminator and generator loss over time**, review the **Loss function** graph.



   The generator loss and discriminator loss are superimposed onto the same graph, but on different scales. The scale of the generator loss is shown on the left side and the scale of the discriminator loss is shown on the right side.

   In this graph, the generator loss plateaus around the 25th epoch, when it stops significantly improving its ability to generate realistic music.

5. Choose **Sample output** to listen to accompaniment tracks that would be generated had inference been performed at that specific epoch.

   For every 50th epoch, you can listen to the accompaniment tracks that could have been generated.

As you can see, both the generator's and discriminator's loss plateaus after the 25th epoch. This plateau indicates that the model was trained successfully.

Evaluating a model when training is unsuccessful

You can determine when training hasn't been successful by evaluating the quality of the sound that your model outputs in the sample output. Notice that in the training, the values for discriminator and generator fluctuate greatly. However, as the training continues, the fluctuations decrease. Towards the end of training, the losses should converge to a value and then should remain at that arbitrary value.

If the learning rate is set too low, the model won't converge. If the learning rate is set too high, the model will train too quickly and will create disarranged sounds. Also, it's important to note that there are different metrics for U-Net and MuseGAN models. Therefore, what works for one model might not have the same effect on another.

Learn more

After creating, training, and evaluating your first custom model, you can continue training more models and composing music. You can train as many models as you like and evaluate them in the music studio.

- Explore the different ways that you can evaluate trained models in AWS DeepComposer (p. 17).
- Open a learning capsule to learn more about different generative AI techniques.
- Try recording a custom input melody (p. 11) and then performing inference with a custom trained model.

Generative techniques in AWS DeepComposer

AWS DeepComposer currently offers two music studio experiences. In both experiences, you can explore the generative models that are available. In this topic, you can learn about the different generative models and how to use them together to create unique compositions.

To start using AWS DeepComposer, you need a *trained model* and an *input melody*. To create a composition, AWS DeepComposer performs inference using the trained model and your input track. During the inference process, the trained model generates a prediction. You can also modify the available inference hyperparameters and model options in the AWS DeepComposer Music studio experiences to fine-tune your musical creation process.

If you are new to machine learning, generative AI, or the different AWS DeepComposer Music studio experiences, see the Getting started with AWS DeepComposer (p. 9) topic, or the tutorial about the basics of generative AI.

For a deeper introduction into the different generative AI techniques supported by AWS DeepComposer, see the **Learning capsules**.

> **Important - Browser requirements**
> AWS DeepComposer fully supports the Chrome browser. Other browsers offer limited support for the AWS DeepComposer console and hardware. For more information about browser compatibility, see *Browser support* (p. 52).

Using the AR-CNN technique in the AWS DeepComposer Music studio

The autoregressive convolutional neural network (AR-CNN) technique is supported in both AWS DeepComposer Music studio experiences, classic and remixed.

The AR-CNN generative technique uses a U-Net architecture originally developed for image generation tasks. In AWS DeepComposer, the AR-CNN learns to compose music by first attempting to detect notes that sound missing or out of place while the model is being trained. Then it replaces those notes with notes it thinks would likely appear in the dataset that it was trained on. The AR-CNN in AWS DeepComposer was trained using a dataset that consists of chorales by Johann Sebastian Bach. To train this model, the audio inputs where first converted to piano roll images. In each piano roll image, the horizontal axis represents time and the vertical axis represents pitch.

Performing inference with the AR-CNN technique in the classic music studio

In the classic music studio experience, you can use a sample melody, record a melody, or import a melody. When you select your input melody in the music studio (p. 10) and choose **Enhance input melody**, the AR-CNN technique modifies the notes in your input melody.

You can use the **Advanced parameters** to adjust how much your input track is modified during inference (p. 3). If you are unhappy with some of the decisions that the model made during inference, you can help the model by editing notes in the melody. Use the **Edit melody** tool to add or remove

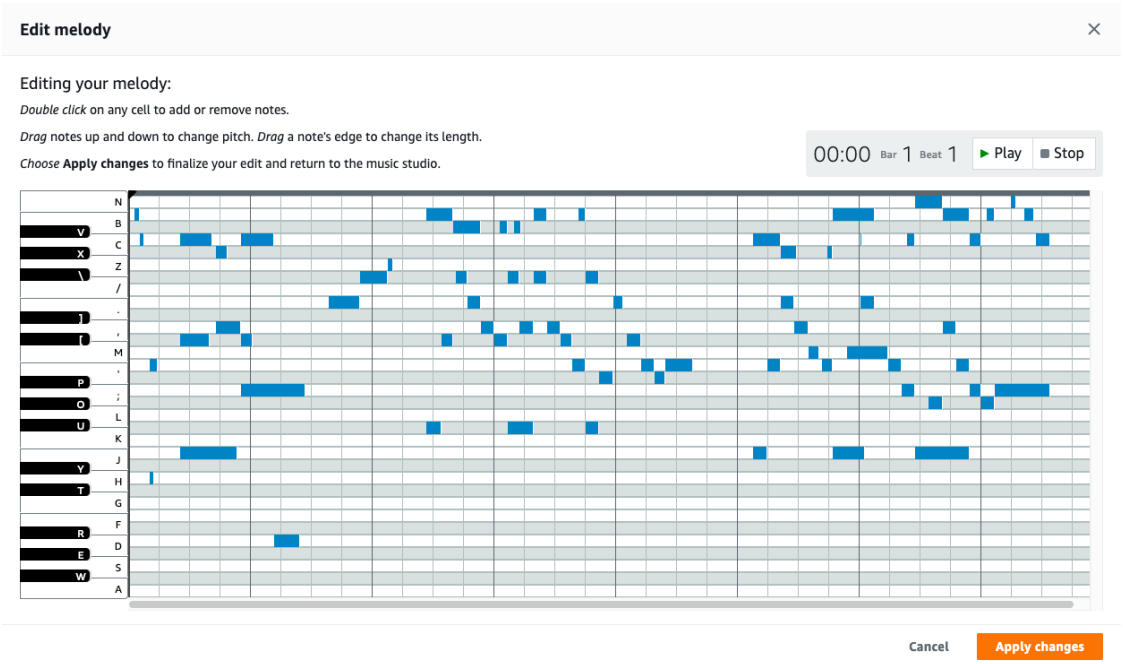
notes, or to change the pitch or the length of notes in the track that was generated. You can then perform inference, again, on the edited track.

Edit melody changes are not saved until inference is performed again.
To save these changes prior to to performing inference again, choose **Download melody**.

To edit your input melody

1. Open the AWS DeepComposer console.
2. In the navigation pane, choose **Music studio**.
3. To open the **Input melody** section, choose the right arrow (▸).
4. Choose **Edit melody**.
5. Under **Source of input melody**, choose **Sample melody**, **Custom recorded melody**, or **Imported track**.
6. On the **Edit melody** page, you can edit your track in the following ways:

   - Choose (double-click) a cell to add or remove a note.
   - Drag a cell up or down to change the pitch of a note.
   - Drag the edge of a cell left or right to change the length of a note.



7. To listen to your changes, choose **Play** (▸).
8. When you have finished, choose **Apply changes**.

Performing inference with the AR-CNN technique in the remixed music studio

In the remixed music studio experience, you can use a sample track, record a custom track, or import a track. You can choose the AR-CNN technique on the **ML technique** page.

The AR-CNN technique modifies the notes in your input melody. After you create your first new melody, you can modify the **AR-CNN parameters** that are available for this technique.

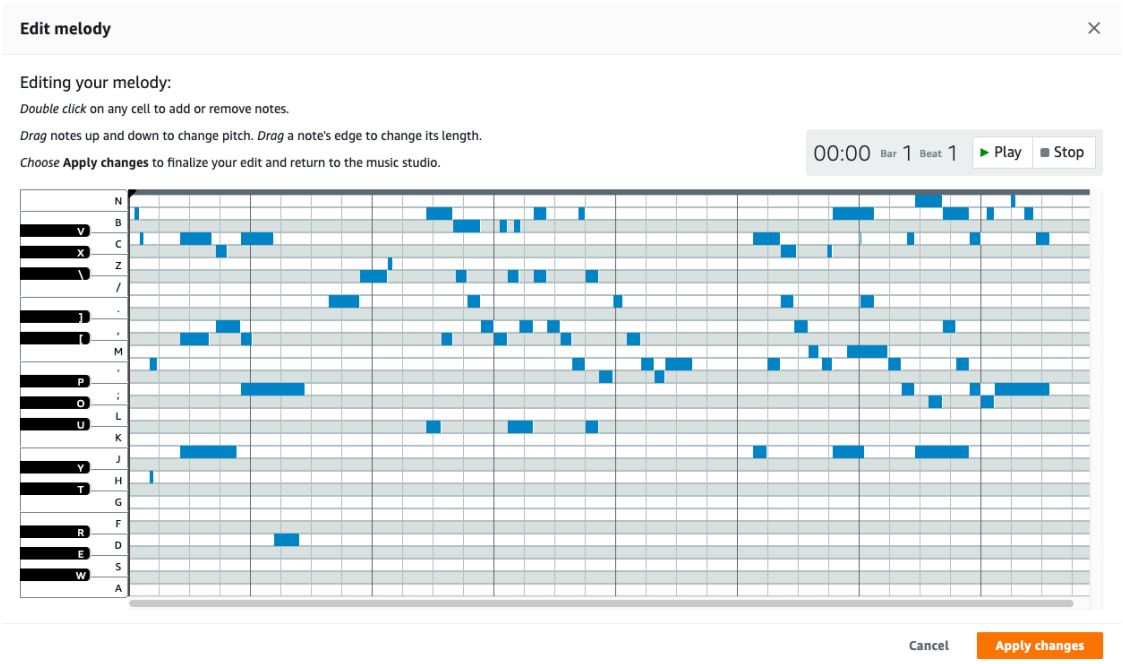
This model adds and removes notes in your input track. If you are unhappy with some of the decisions that the model made during inference (p. 3), you can help the model by editing the notes in the melody. Use the **Edit melody** tool to add or remove notes, or to change the pitch or the length of notes in the track that was generated. You can then perform inference on the edited track.

On the **Inference output** page, you can choose **Enhance again** to perform inference multiple times. You can also access the **Edit melody** tool from this page.

> **Edit melody changes are not saved until inference is performed again.**
> To save these changes prior to to performing inference again, choose **Download melody**.

To edit your input melody

1. Choose **Edit melody**.
2. On the **Edit melody** page, you can edit your track in the following ways:
   - Choose (double-click) a cell to add or remove a note.
   - Drag a cell up or down to change the pitch of a note.
   - Drag the edge of a cell left or right to change the length of a note.



3. To listen to your changes, choose **Play** (▶).
4. When you have finished, choose **Apply changes**.

Model parameter options for the AR-CNN technique

The model parameters for the AR-CNN technique are the same in both AWS DeepComposer Music studio experiences.

The AR-CNN technique uses the **AutoregressiveCNN Bach** sample model. To help create unique musical tracks, you can modify the different parameters. These parameters, also known as *inference hyperparameters*, control how much the model changes your melody.

Sampling iterations

Controls the number of times your input melody is passed through the model. Increasing the number of iterations results in more notes being added and removed from the melody.

Maximum input notes to remove

Controls the *percentage* of input melody to be removed during inference. By increasing this parameter, you are allowing the model to use less of the input melody as a reference during inference. After performing inference, you can use the **Edit melody (p. 36)** tool to further modify your melody.

Maximum number of notes to add

Controls the *number of notes* that can be added to the input melody. By increasing this number, you might introduce notes which sound out of place into your melody. It's also a creative way to experiment with your chosen melody. After performing inference, you can use the **Edit melody (p. 36)** tool to further modify your melody.

Creative risk

Controls how much the model can deviate from the music that it was trained on. More technically, when you change this value, you are changing the shape of the output probability distribution. If you set this value too low, the model will choose only high-probability notes. If you set this value too high, the model will more likely choose lower-probability notes.

How the AR-CNN technique works with other generative techniques in the music studio

In both AWS DeepComposer Music studio experiences, you can use the AR-CNN technique in collaboration with the GANs technique or the Transformers technique. When using the AR-CNN technique with either technique, you must start by working with the AR-CNN technique.

With the GANs technique, you can create accompaniment tracks, but when you return to your edited melody, those accompaniment tracks will no long be available.

After you switch to the Transformers technique, you cannot return to the AR-CNN technique to further edit your melody.

Note
Each time your perform inference with the AR-CNN technique, your new composition is saved automatically. If you want to save a melody that you have modified using the **Edit melody** tool prior to performing inference another time, choose **Download melody**.

Using the GANs technique in the AWS DeepComposer Music studio

The generative adversarial networks (GANs) technique is supported in both AWS DeepComposer Music studio experiences, remixed and classic.

GANs are neural networks that consist of a *generator* and a *discriminator*. In AWS DeepComposer, the generator learns to compose music that is as realistic as possible with feedback from the discriminator. The discriminator treats the generator's output as being as unrealistic as possible while holding the input training sample as the ground truth.

In the music studio, two different GAN architectures are available, MuseGAN and U-Net. Both architectures use a convolutional neural network (CNN) because the first step in training these models is to convert the input audio into an image-based representation of music called a piano roll.

MuseGAN

The MuseGAN architecture was built specifically to generate music. AWS DeepComposer comes with five genre-based sample models that use the MuseGAN architecture. In AWS DeepComposer, you can also train a custom MuseGAN model and use it in either music studio experience.

U-Net

The U-Net architecture has been adapted for music generation. It was originally developed for image-generation tasks. The name for this architecture stems from its unique U shape. This shape allows the CNN to pass information on the left side (the encoder) to the layers on the right side (the decoder) without passing through the entire neural network. AWS DeepComposer does not have a sample model that uses the U-Net architecture. Instead, you can train a custom U-Net model and use it in either music studio experience.

To learn more about training a custom model using AWS DeepComposer, refer to the topic on training a custom model (p. 15).

Performing inference with the GANs technique in the AWS DeepComposer Music studio experiences

In either AWS DeepComposer Music studio experience, you can use the GANs technique to perform inference, which creates new accompaniment tracks for your composition.

You can use a sample melody, record a custom melody, or import a melody. For more information, see the topic on creating compositions with the AWS DeepComposer Music studio (p. 10).

The GANs technique creates four accompaniment tracks based on your input melody. Each accompaniment track initially comes from a broad class of musical instruments. You can modify the instrument class and enter for each accompaniment track except for drums.

To modify your accompaniment tracks, you can use the same steps in either AWS DeepComposer Music studio experience. The only difference is where this feature is located. In the classic music studio experience, you can access your newly generated tracks on the music studio landing page. In the remixed music studio experience, you can access the accompaniments on the **Inference output** page.

To change an instrument type after generating a composition

1. To open an **Instrument type**, such as **String ensemble 1**, choose the right arrow (▸).
2. Choose an instrument type.
3. Choose the down arrow (▾) to open the **Instrument** menu.
4. Choose an instrument.

Model parameter options for the GANs technique

When using the GANs generative technique, you can select from either **Sample models** or **Custom models**. In either music studio experience, you can choose from five different genre-based **Sample models**, which were trained using the MuseGAN architecture.

If you trained a custom U-Net or MuseGAN model, you can find it and the **Sample models** under **Model**.

In the remixed music studio, you can find the **Model** option on both the **ML technique** and the **Inference output** pages.

How the GANs technique works with other generative techniques in the music studio

In either AWS DeepComposer Music studio experience, you can use the GANs technique in collaboration with the AR-CNN technique. The GANs technique is not compatible with the Transformers technique.

To use the GANs technique with the AR-CNN technique, you must first use the AR-CNN technique to enhance your input melody. When ready, you can switch to the GAN technique and create accompaniment tracks.

> **Note**
> In the classic music studio experience, you can return to the AR-CNN technique after creating accompaniment tracks with the GANs technique. If you do so, the tracks that were generated with the GANs technique will be automatically saved as a new composition. When you modify your input melody using the AR-CNN technique, you must generate new accompaniment tracks, because the previously created accompaniment tracks cannot be accessed.

Using the Transformers technique in the AWS DeepComposer Music studio

The Transformers generative technique is used to solve sequence modeling problems. In sequential modeling, the model takes into account previous outputs when generating the next output. To do this, the Transformers technique uses the concept of attention. This concept allows the model, when given a specific input sequence, to better understand which other parts of the sequence are important.

Unlike the GANs and AR-CNN techniques, the Transformers technique doesn't treat music generation as an image-generation problem. Instead, it treats music generation like a text-generation problem. To solve this problem, the model needs to create tokens that represent the musical inputs. The tokens are used when predicting which notes should come next. In AWS DeepComposer, the Transformers technique uses the style, pattern, or musical motifs found in the input melody when generating the extended output track.

> **Edit melody changes are not saved until inference is performed again.**
> To save these changes prior to to performing inference again, choose **Download melody**.

Performing inference with the Transformers technique

To get started with the Transformers technique, we recommend using one of the **Sample melodies**, specifically one **Recommended for the Transformers technique**. These options represent the complex classical melodies that work best with the model. After selecting your input melody, you can create your first composition by choosing **Extend input melody**. The model extends your input melody by up to 30 seconds. Similar to when you work with the AR-CNN technique, you can, after performing inference, use the **Edit melody (p. 28)** tool to add or remove notes, or to change the pitch or the length of notes in the extension that was generated.

Unlike the AR-CNN technique, the Transformers technique doesn't modify the changes that you make to the input melody when you perform inference again. Making changes to the input melody will influence the extension that is generated by the Transformers technique.

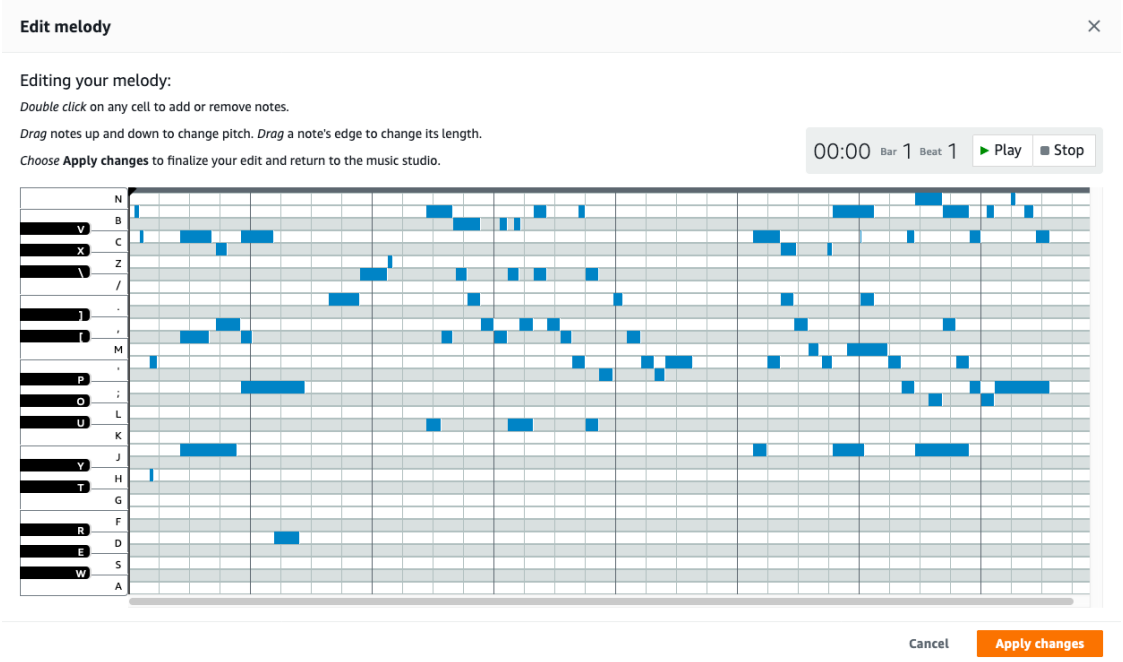
Using the Edit melody tool with the Transformers technique in the classic music studio

You can find the **Edit melody** tool on the landing page.

To edit your input melody

1. Open the AWS DeepComposer console.
2. In the navigation pane, choose **Music studio**.
3. To open the **Input melody** section, choose the right arrow (▸).
4. Under **Source of input melody**, choose **Sample melody**, **Custom recorded melody**, or **Imported track**.
5. Choose **Edit melody**.
6. On the **Edit melody** page, you can edit your track in the following ways:

   - Choose (double-click) a cell to add or remove notes.
   - Drag a cell up or down to change the pitch of a note.
   - Drag the edge of a cell left or right to change the length of a note.



7. To listen to your changes, choose **Play** (▸).
8. When you have finished, choose **Apply changes**.

Edit melody changes are not saved until inference is performed again.
To save these changes prior to to performing inference again, choose **Download melody**.

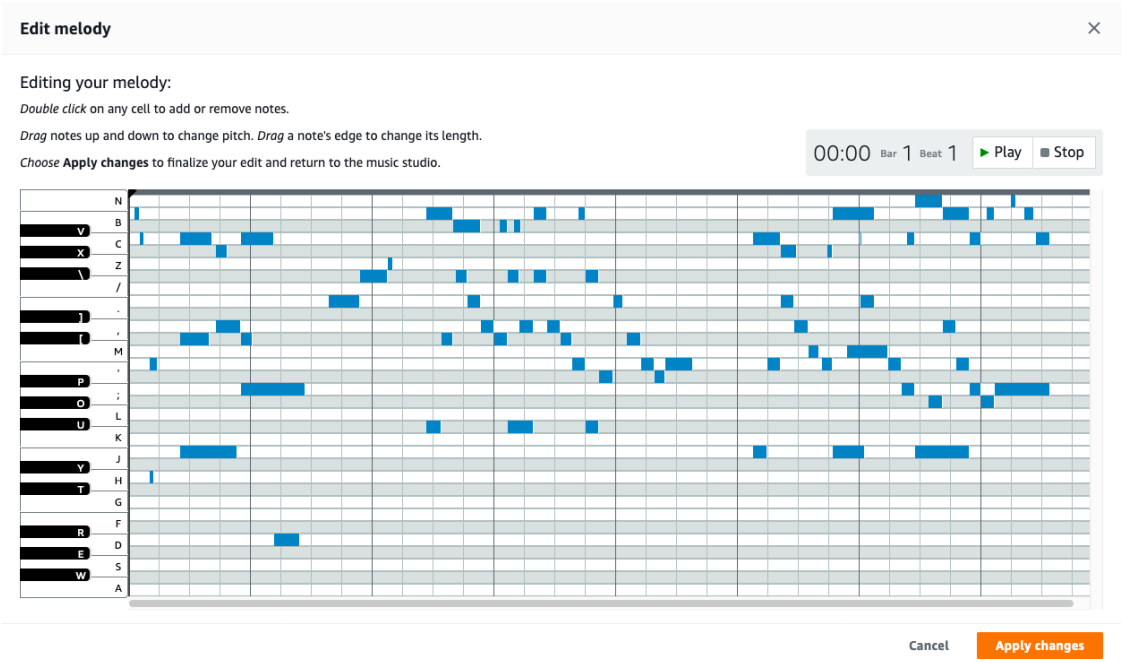
Using the Edit melody tool with the Transformers technique in the remixed music studio

You can find the **Edit melody** tool on the **Input track**, **Inference output**, and **Next steps** pages.

To edit your melody in the remixed music studio experience

1. Choose **Edit melody**.
2. On the **Edit melody** page, you can edit your track in the following ways:

   - Choose (double-click) a cell to add or remove notes.
   - Drag a cell up or down to change the pitch of a note.
   - Drag the edge of a cell left or right to change the length of a note.



3. To listen to your changes, choose **Play** (▶).
4. When you have finished, choose **Apply changes**.

Edit melody changes are not saved until inference is performed again.
To save these changes prior to to performing inference again, choose **Download melody**.

Model parameter options for the Transformers technique

To create unique sounding musical creations, you can modify the available inference hyperparameters. In the classic music studio experience these are called **Advanced parameters** and in the remixed music studio experience they are called **Tranformers parameters**. These parameters control how your input track is modified. These parameters can be broadly categorized in three different groups.

Group 1

Modifying these parameters directly affects how inference is performed on your input melody. As mentioned previously, during model training, the musical inputs are converted into tokens. The sum of available tokens represents the total musical knowledge learned during training. Contained within each token is also a probability.

Sampling technique

The Transformers technique in AWS DeepComposer supports three different sampling techniques. The sampling technique determines how the new melody notes are chosen.

- **TopK**: The next note is chosen by limiting the list of next available notes based on the value selected for **Sampling threshold**. Then the next note is chosen from that new list.
- **Nucleus**: The next note is chosen by limiting the list of next available notes based on the value selected for **Sampling threshold**. The value set for the **Sampling threshold** represents the maximum allowable cumulative sum of the token's individual probabilities when ranked greatest to least. The next note is chosen from that new list of available tokens.
- **Random**: Unlike the previous two techniques, when **Random** is selected, the list of available tokens isn't modified. Instead, the model can choose from any available note at any point while inference is being performed.

Sampling threshold

The threshold sets the number of available notes that the model can choose from during inference. The value selected changes based on the **Sampling technique** selected. When the **Random** technique is selected, this parameter is not available.

Creative risk

Increasing this value allows the model to deviate from the music it was trained on, and the generated music sounds more experimental.

Setting the Sampling technique equal to TopK and the Sampling threshold equal to 0.80

This sampling technique uses the total sum of available tokens learned during training to predict an upcoming note. When a new note needs to be predicted, the total number of tokens are ranked numerically from greatest to least based on their probability at that moment. When you modify the **Sampling threshold**, you are changing the number of notes from which the model can choose.

For example, the model that AWS DeepComposer uses, TransformerXLClassical, learned 310 tokens during training. Setting the **Sampling threshold** to 0.80 means that the model can randomly choose from, *310 tokens x 0.80*, the top 248 tokens during inference. Setting the **Sampling threshold** to values closer to the maximum input value (0.99) means that the model has a greater chance of choosing lower-probability tokens. Setting the values close to the minimum input value (0.1) means that the model is more restricted to higher-probability tokens.

Setting the Sampling technique equal to Nucleus and the Sampling threshold equal to 0.96

This sampling techniqueuses the probabilities associated with the tokens to predict the upcoming note. When a new note needs to be predicted, all 310 tokens are ranked from greatest to least using their probabilities. When you modify the **Sampling threshold**, you are setting the threshold equal to the maximum allowable cumulative sum of the ranked probabilities.

For example, imagine at the top of your descending list of token probabilities are the following five probabilities: `[0.4, 0.3, 0.2, 0.05, 0.05]`. If you take the cumulative sum of these probabilities, you end up with this list: `[0.4, 0.7, 0.9, 0.95, 1.0]`. In this case, each probability represents a note (which is represented by a token). Setting the **Sampling threshold** to 0.96 means that, at that moment, the model can pick a note randomly from this remaining list of partially summed probabilities, `0.4, 0.7, 0.9, 0.95]`, which represents the original list of ranked probabilities, `[0.4, 0.3, 0.2, 0.05]`. So, in the moment, the model can pick from two higher-probability notes, 0.4 and 0.3, a slightly lower-probability note, 0.2, and a low-probability note, 0.05.

Group 2

These parameters tell the model how much new track it should attempt to generate and what portion of the input track should be used during inference.

Input duration

This parameter selects the portion of your track, counting in seconds from the end, that should be used for inference.

Track extension duration

This parameter selects the amount of time, in seconds, the model will attempt to generate.

Group 3

During inference, the Transformers model can create musical artifacts. Both of the following parameters can help remove either long periods of silence or moments when a note is being held for an expectedly long period of time.

Maximum rest time

Silence is compressed when this value is exceeded.

Maximum note length

Held notes are compressed when this value is exceeded.

How the Transformers technique works with other generative techniques in the music studio

Use the Transformers technique as either the last step or the only step in your music creation process. To use the AR-CNN technique with the Transformers technique, you must start with the AR-CNN technique. Because the Transformers technique extends your input melody, the output is not compatible with the GANs technique.

Launching an Amazon SageMaker notebook instance

To launch an AWS DeepComposer GitHub-based project, use an SageMaker notebook instance. A SageMaker notebook instance is a machine learning (ML) compute instance that runs the Jupyter Notebook App. SageMaker manages creating the instance and all of the resources required to use it. Jupyter notebooks are open-source web applications that you can use to create and share documents that contain live code, equations, visualizations, and instructive text. Use Jupyter notebooks in your notebook instance to prepare and process data, write code to train models, deploy models to SageMaker, and test or validate your models.

> **Note**
> To use the AWS DeepComposer console and other AWS services, you need an AWS account. If you don't have an account, go to aws.amazon.com and choose **Create an AWS Account**. For detailed instructions, see Create and Activate an AWS Account.
> As a best practice, you should also create an AWS Identity and Access Management (IAM) user with administrator permissions and use that for all work that does not require root credentials. Create a password for console access, and access keys to use command line tools. For more information, see Creating Your First IAM Admin User and Group in the *IAM User Guide*.

Topics

Request a Service limit increase for use with a Amazon SageMaker Notebook Instance

In order to use a compute enhanced notebook instance, you must submit a request for a service limit increase to the AWS Support Center.

1. Open the AWS Support Center console.
2. On the **AWS Support Center** page, choose **Create Case** and then choose **Service limit increase**.
3. In the **Case classification** panel under **Limit type**, search for SageMaker;.
4. In the **Request** panel, choose the **Region** that you are working in. For **Resource Type**, choose **SageMaker Notebooks**.
5. For **Limit** choose **ml.c5.4xlarge** instances.
6. For **New Limit Value**, verify that the value is **1**.
7. In **Case description**, provide a brief explanation of why you need the **Service limit increase**. For example, I need to use this compute optimized notebook instance to train a deep learning model using TensorFlow.
8. In **Contact options**, provide some details about how you would like to be contacted by the AWS service support team on the status of your **Service limit increase** request.
9. Choose **submit**.

Create a SageMaker notebook instance and launch a Jupyter notebook

To create a notebook instance and launch a Jupyter notebook

1. Sign in to the Amazon SageMaker console.
2. In the navigation pane, choose **Notebook instances**.
3. On the **Notebook instances** page, choose **Create notebook instance**.
4. On the **Create notebook instance** page, for **Notebook instance name**, enter a name for the notebook instance.
5. For **Notebook instance type**, choose an `ml.p3.2xlarge` instance.
6. Under **Permissions and encryption**Choose an **IAM role** to set up access permissions and encryption.
   - If you already have a SageMaker IAM role, choose it from the list.
   - If you're to SageMaker, create an IAM role by choosing **Create a new role**. On the **Create an IAM role** page, choose **Any S3 bucket** to give your role access to your S3 bucket. Choose **Create Role**.
   - Make sure **Enable - Give users root access to the notebook** is selected from the **Root access** menu.
7. Choose **Create role**.
8. On the **Create a notebook instance** page, choose **IAM role**, and then choose your IAM role from the list.
9. Open the **GitHub repositories** panel.
10. For **Default repository**, choose **Clone a public Git repository to this notebook instance only**.
11. The Git repository called AR-CNN contains the Jupyter notebook required for this custom project. Copy this link and paste it into the field under **Github repositories**.
12. Choose **Create notebook instance**.
13. On the **Notebook instances** page, choose **Open Jupyter** to launch your new Jupyter notebook.

Editing music with AWS DeepComposer

To edit sample, imported, and custom recorded melodies, use the AWS DeepComposer Music studio.

Currently, AWS DeepComposer provides two different music studio experiences, classic and remixed. Some editing tools are not supported in the remixed music studio experience, but you can access them in the classic music studio.

Available editing tools

Each music studio experience has a different set of editing tools available.

Some tools are unavailabe in the remixed music studio experience, to edit your tempo or pitch use the classic music studio experience.

Editing tool	Music studio (classic)	Music studio (remixed)
Tempo	✓	⚠
Pitch	✓	⚠
Edit melody	✓	✓

Use the following steps to edit your melody no matter which music studio experience you chose.

Changing tempo

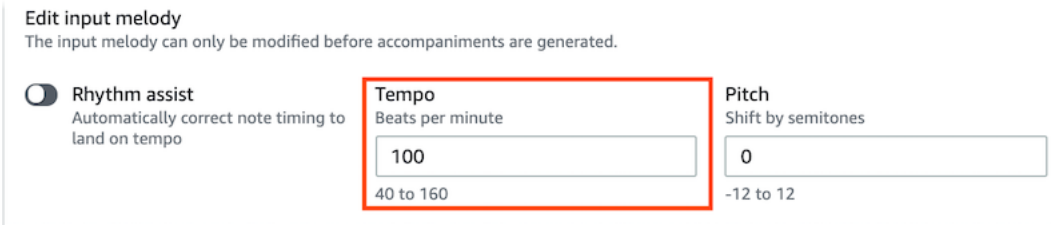
Tempo determines how fast music is played. Music typically follows a certain beat or meter, which drives the rhythm of the notes. The speed of this beat is measured in beats per minute. A higher number of beats per minute corresponds to a faster tempo, or playback speed.

You can use the classic music studio to change the tempo for a composition you created in the remixed music studio.

Changing tempo in the classic music studio experience

To change the tempo of an input melody

1. Open the AWS DeepComposer console.
2. In the navigation pane, choose **Music studio**.
3. To open the **Input melody** section, choose the right arrow (▸).
4. For **Input melody**, choose the melody whose tempo you want to change.
5. For **Tempo**, enter a value of 40–160 beats per minute. Your change is saved automatically.



6. To listen to your change, choose **Play** (▸).

Changing tempo in the remixed music studio experience

To change the tempo of compositions that you created in the remixed music studio experience, use the classic music studio experience.

To change the tempo of an input melody

1. Open the AWS DeepComposer console.
2. In the navigation pane, choose **Compositions**.
3. Select the composition whose tempo you want to change.
4. To load your composition in the classic music studio experience, choose **Load into Music studio**.
5. To open the **Input melody** section, choose the right arrow (▸).
6. For **Tempo**, enter a value of 40–160 beats per minute. Your change is saved automatically.
7. To listen to your change, choose **Play** (▸).

Changing pitch

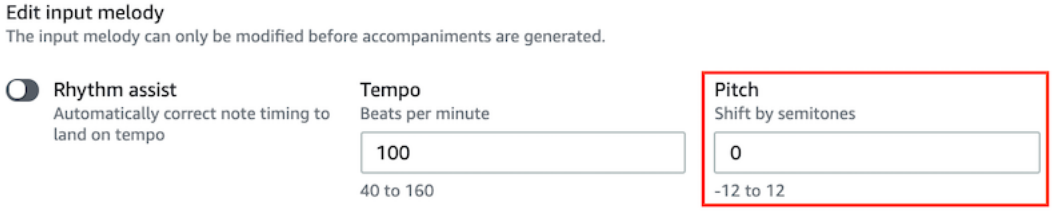
Pitch controls how low or high a note sounds. In AWS DeepComposer, you can adjust the pitch of an input melody by semitones. A *semitone* is a half step, and is the smallest interval used in western music. You can adjust pitch down 12 semitones (-12) or up 12 semitones (12).

You can use the classic music studio to change the pitch of a composition you created in the remixed music studio.

Changing pitch in the classic music studio experience

To change the pitch of an input melody

1. Open the AWS DeepComposer console.
2. In the navigation pane, choose **Music studio**.
3. To open the **Input melody** section, choose the right arrow (▸).
4. For **Input melody**, choose the melody whose pitch you want to change.
5. For **Pitch**, enter a value of -12 to 12 semitones. Your change is saved automatically.



6. To listen to your change, choose **Play** (▸).

Changing the pitch in the remixed music studio experience

To change the pitch of compositions that you created in the remixed music studio experience, use the classic music studio experience.

To change the pitch of an input track

1. Open the AWS DeepComposer console.
2. In the navigation pane, choose **Compositions**.
3. Select the composition whose pitch you want to change.
4. To load your composition in the classic music studio experience, choose **Load into Music studio**.
5. To open the **Input melody** section, choose the right arrow (▸).
6. For **Pitch**, enter a value of -12 to 12 semitones. Your change is saved automatically.
7. To listen to your change, choose **Play** (▸).

Edit the melody

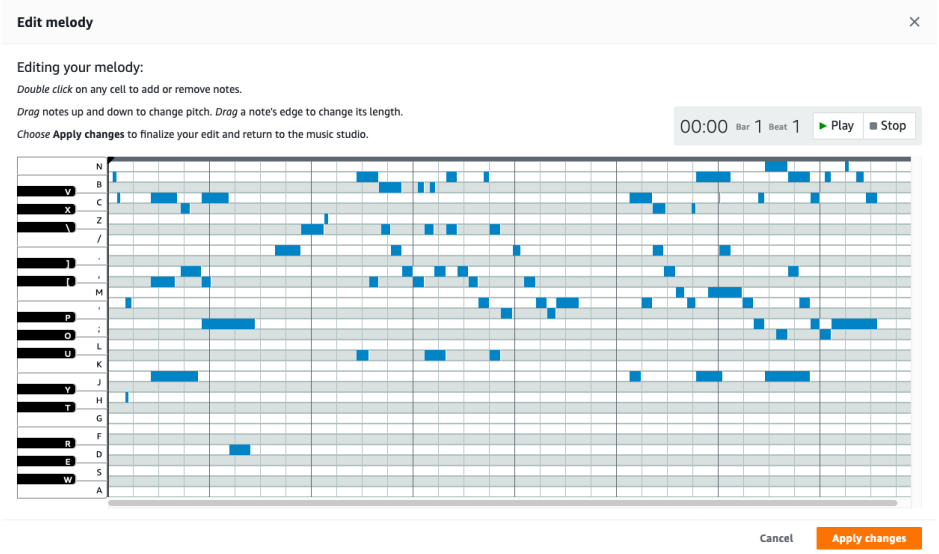
You can edit your input melody when you create compositions in the AWS DeepComposer Music studio. The **Edit melody** tool can be used on any input track.

The **Edit melody** tool is available in both classic and remixed music studio experiences. Use the sections below to help you locate and use the tool based on the music studio experience you are using.

> **Edit melody changes are not saved until inference is performed again.**
> To save these changes prior to to performing inference again, choose **Download melody**.

Using the **Edit melody** tool in the classic music studio

When you use the **Edit melody** tool, your changes are *not* automatically saved. To save your changes, you must perform inference again no matter which ML technique you used.



To edit your input melody

1. Open the AWS DeepComposer console.
2. In the navigation pane, choose **Music studio**.
3. To open the **Input melody** section, choose the right arrow (▸).
4. Choose **Edit melody**.
5. On the **Edit melody** page, you can edit your track in the following ways:
   - Choose (double-click) a cell to add or remove notes.
   - Drag a cell up or down to change the pitch of a note.
   - Drag the edge of a cell left or right to change the length of a note.
6. To listen to your changes, choose **Play** (▸).
7. When you have finished, choose **Apply changes**.

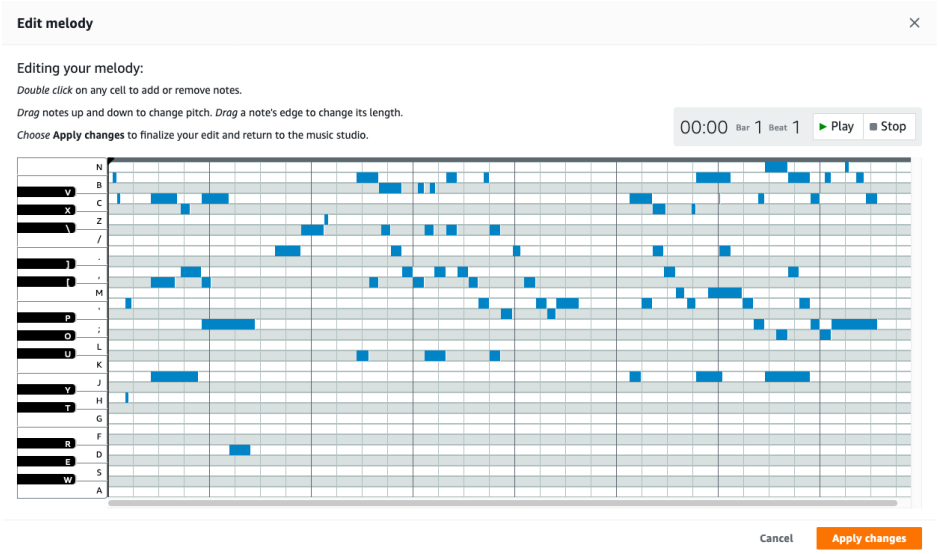
Edit melody changes are not saved until inference is performed again.
To save these changes prior to to performing inference again, choose **Download melody**.

Using the **Edit melody** tool in the remixed music studio

In the remixed music studio experience, you can use the **Edit melody** tool on the **Input track**, **Inference output**, and **Next steps** pages.

Although the **Edit melody** tool is availabe on the **Next steps** page, use the To edit your input track on the **Next steps** page (p. 38) procedure to either save or download your changes.

Edit melody changes are not saved until inference is performed again.
To save these changes prior to to performing inference again, choose **Download melody**.



To edit your input track on the Input track, and Inference output pages

1. In the remixed music studio, on either the **Input track** or **Inference output** pages.
2. Choose **Edit melody**.
3. On the **Edit melody** page you can edit your track in the following ways:
   - Choose (Double-click) on a cell to add or remove notes.

- Drag a cell up or down to change the note's pitch.
- Drag the edge of a cell left or right to change the length of a note.

4. To listen to your changes, choose **Play** (▶).
5. When finished, choose **Apply changes**.
6. (Optional) Choose **Download melody** to save your changes if you do not want to perform inference again.

To edit your input track on the Next steps page

1. In the remixed music studio, on the **Next steps** page.
2. Choose **Edit melody**.
3. On the **Edit melody** page you can edit your track in the following ways:

   - Choose (Double-click) on a cell to add or remove notes.
   - Drag a cell up or down to change the note's pitch.
   - Drag the edge of a cell left or right to change the length of a note.

4. To listen to your changes, choose **Play** (▶).
5. When finished, choose **Apply changes**.
6. (Optional) Choose **Download melody** to save your changes if you do not want to perform inference again.
7. (Optional) To perform inference again to save your changes made with **Edit melody** tool use the steps below

   a. In the primary navigation pane, choose **3. Inference output**.
   b. Based on the ML technique previsouly selected choose either, **Extend again** or **Enhance again** to perform inference again.
   c. When finished, choose **Continue** to return to the **Next steps** page.

Security in AWS DeepComposer

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from data centers and network architectures that are built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS Compliance Programs. To learn about the compliance programs that apply to AWS DeepComposer, see AWS Services in Scope by Compliance Program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using AWS DeepComposer. The following topics show you how to configure AWS DeepComposer to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your AWS DeepComposer resources.

Topics

AWS Identity and Access Management for AWS DeepComposer

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use DeepComposer resources. IAM is an AWS service that you can use with no additional charge.

Topics

- Troubleshooting AWS DeepComposer AWS Identity and Access Management (p. 47)

Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work that you do in DeepComposer.

Service user – If you use the DeepComposer service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more DeepComposer features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator. If you cannot access a feature in DeepComposer, see Troubleshooting AWS DeepComposer AWS Identity and Access Management (p. 47).

Service administrator – If you're in charge of DeepComposer resources at your company, you probably have full access to DeepComposer. It's your job to determine which DeepComposer features and resources your employees should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM. To learn more about how your company can use IAM with DeepComposer, see How AWS DeepComposer works with IAM (p. 43).

IAM administrator – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to DeepComposer. To view example DeepComposer identity-based policies that you can use in IAM, see AWS DeepComposer identity-based policy examples (p. 45).

Authenticating With Identities

Authentication is how you sign in to AWS using your identity credentials. For more information about signing in using the AWS Management Console, see Signing in to the AWS Management Console as an IAM user or root user in the *IAM User Guide*.

You must be *authenticated* (signed in to AWS) as the AWS account root user, an IAM user, or by assuming an IAM role. You can also use your company's single sign-on authentication or even sign in using Google or Facebook. In these cases, your administrator previously set up identity federation using IAM roles. When you access AWS using credentials from another company, you are assuming a role indirectly.

To sign in directly to the AWS Management Console, use your password with your root user email address or your IAM user name. You can access AWS programmatically using your root user or IAM users access keys. AWS provides SDK and command line tools to cryptographically sign your request using your credentials. If you don't use AWS tools, you must sign the request yourself. Do this using *Signature Version 4*, a protocol for authenticating inbound API requests. For more information about authenticating requests, see Signature Version 4 signing process in the *AWS General Reference*.

Regardless of the authentication method that you use, you might also be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see Using multi-factor authentication (MFA) in AWS in the *IAM User Guide*.

AWS account root user

When you first create an AWS account, you begin with a single sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you do not use the root user for your everyday tasks, even the administrative ones. Instead, adhere to the best practice of using the root user only to create your first IAM user. Then securely lock away the root user credentials and use them to perform only a few account and service management tasks.

IAM Users and Groups

An *IAM user* is an identity within your AWS account that has specific permissions for a single person or application. An IAM user can have long-term credentials such as a user name and password or a set of access keys. To learn how to generate access keys, see Managing access keys for IAM users in the *IAM User Guide*. When you generate access keys for an IAM user, make sure you view and securely save the key pair. You cannot recover the secret access key in the future. Instead, you must generate a new access key pair.

An *IAM group* is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named *IAMAdmins* and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see When to create an IAM user (instead of a role) in the *IAM User Guide*.

IAM Roles

An *IAM role* is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. You can temporarily assume an IAM role in the AWS Management Console by switching roles. You can assume a role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see Using IAM roles in the *IAM User Guide*.

IAM roles with temporary credentials are useful in the following situations:

- **Temporary IAM user permissions** – An IAM user can assume an IAM role to temporarily take on different permissions for a specific task.
- **Federated user access** – Instead of creating an IAM user, you can use existing identities from AWS Directory Service, your enterprise user directory, or a web identity provider. These are known as *federated users*. AWS assigns a role to a federated user when access is requested through an identity provider. For more information about federated users, see Federated users and roles in the *IAM User Guide*.
- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see How IAM roles differ from resource-based policies in the *IAM User Guide*.
- **Cross-service access** – Some AWS services use features in other AWS services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.
  - **Principal permissions** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. Policies grant permissions to a principal. When you use some services, you might perform an action that then triggers another action in a different service. In this case, you must have permissions to perform both actions. To see whether an action requires additional dependent actions in a policy, see Actions, Resources, and Condition Keys for AWS DeepComposer in the *Service Authorization Reference*.
  - **Service role** – A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.
  - **Service-linked role** – A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear

in your IAM account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see Using an IAM role to grant permissions to applications running on Amazon EC2 instances in the *IAM User Guide*.

To learn whether to use IAM roles or IAM users, see When to create an IAM role (instead of a user) in the *IAM User Guide*.

Managing Access Using Policies

You control access in AWS by creating policies and attaching them to IAM identities or AWS resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. You can sign in as the root user or an IAM user, or you can assume an IAM role. When you then make a request, AWS evaluates the related identity-based or resource-based policies. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

Every IAM entity (user or role) starts with no permissions. In other words, by default, users can do nothing, not even change their own password. To give a user permission to do something, an administrator must attach a permissions policy to a user. Or the administrator can add the user to a group that has the intended permissions. When an administrator gives permissions to a group, all users in that group are granted those permissions.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

Identity-Based Policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Creating IAM policies in the *IAM User Guide*.

Identity-based policies can be further categorized as *inline policies* or *managed policies*. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see Choosing between managed policies and inline policies in the *IAM User Guide*.

Resource-Based Policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform

on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

Access Control Lists (ACLs)

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see Access control list (ACL) overview in the *Amazon Simple Storage Service Developer Guide*.

Other Policy Types

AWS supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see How SCPs work in the *AWS Organizations User Guide*.
- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see Session policies in the *IAM User Guide*.

Multiple Policy Types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy evaluation logic in the *IAM User Guide*.

How AWS DeepComposer works with IAM

Before you use IAM to manage access to AWS DeepComposer, you should understand which IAM features are available to use with AWS DeepComposer. To get a high-level view of how AWS DeepComposer and other AWS services work with IAM, see AWS Services That Work with IAM in the *IAM User Guide*.

Topics

AWS DeepComposer identity-based policies

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. DeepComposer supports specific actions, resources, and condition keys. To learn about all of the elements that you use in a JSON policy, see IAM JSON Policy Elements Reference in the *IAM User Guide*.

Actions

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Action` element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Policy actions usually have the same name as the associated AWS API operation. There are some exceptions, such as *permission-only actions* that don't have a matching API operation. There are also some operations that require multiple actions in a policy. These additional actions are called *dependent actions*.

Include actions in a policy to grant permissions to perform the associated operation.

Resources

AWS DeepComposer supports specifying resource Amazon Resource Name (ARNs) in a policy. To restrict access to AWS DeepComposer resources, you can use authorization based on AWS DeepComposer tags.

For more information about ARNs, see Amazon Resource Names (ARNs) and AWS Service Namespaces.

Condition keys

AWS DeepComposer does not support specifying condition keys.

To see all AWS global condition keys, see AWS Global Condition Context Keys in the *IAM User Guide*.

Examples

To view examples of DeepComposer identity-based policies, see AWS DeepComposer identity-based policy examples (p. 45).

Authorization based on AWS DeepComposer tags

You can attach tags to AWS DeepComposer resources.

You can use conditions in your identity-based policy to control access to AWS DeepComposer resources based on tags.

To control access based on tags, you provide tag information in the condition element of a policy using the `AWS DeepComposer:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`*, or `aws:TagKeys` condition keys.

For more information about how to tag AWS DeepComposer resources in the AWS DeepComposer console, see Tagging with AWS DeepComposer (p. 53).

The following example is an identity-based policy for limiting access to a resources based on the tags on that resource.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
```

```
            "Action": [
                "deepcomposer:ListTagsForResource",
                "deepcomposer:TagResource",
                "deepcomposer:DeleteComposition"
            ],
            "Resource": "*"
        },
        {
            "Sid": "VisualEditor2",
            "Effect": "Allow",
            "Action": [
                "deepcomposer:CreateComposition"
            ],
            "Resource": "*",
            "Condition": {
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": [
                        "exampleKey"
                    ]
                }
            }
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "deepcomposer:GetComposition",
                "deepcomposer:UntagResource"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/exampleKey": "exampleValue"
                }
            }
        }
    ]
}
```

AWS DeepComposer identity-based policy examples

By default, IAM users and roles don't have permission to create or modify DeepComposer resources. They also can't perform tasks using the AWS Management Console, AWS CLI, or AWS API. An IAM administrator must create IAM policies that grant users and roles permission to perform specific API operations on the specified resources they need. The administrator must then attach those policies to the IAM users or groups that require those permissions.

To learn how to create an IAM identity-based policy using these example JSON policy documents, see Creating Policies on the JSON Tab in the *IAM User Guide*.

Topics

Policy best practices

Identity-based policies are very powerful. They determine whether someone can create, access, or delete DeepComposer resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started using AWS managed policies** – To start using DeepComposer quickly, use AWS managed policies to give your employees the permissions they need. These policies are already available in your account and are maintained and updated by AWS. For more information, see Get started using permissions with AWS managed policies in the *IAM User Guide*.
- **Grant least privilege** – When you create custom policies, grant only the permissions required to perform a task. Start with a minimum set of permissions and grant additional permissions as necessary. Doing so is more secure than starting with permissions that are too lenient and then trying to tighten them later. For more information, see Grant least privilege in the *IAM User Guide*.
- **Enable MFA for sensitive operations** – For extra security, require IAM users to use multi-factor authentication (MFA) to access sensitive resources or API operations. For more information, see Using multi-factor authentication (MFA) in AWS in the *IAM User Guide*.
- **Use policy conditions for extra security** – To the extent that it's practical, define the conditions under which your identity-based policies allow access to a resource. For example, you can write conditions to specify a range of allowable IP addresses that a request must come from. You can also write conditions to allow requests only within a specified date or time range, or to require the use of SSL or MFA. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.

Using the DeepComposer console

To access the AWS DeepComposer console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the DeepComposer resources in your AWS account. If you create an identity-based policy that is more restrictive than the minimum required permissions, the console does not function as intended for entities (IAM users or roles) with that policy.

To ensure that those entities can still use the DeepComposer console, also attach the following AWS managed policy to the entities. For more information, see Adding Permissions to a User in the *IAM User Guide*.

```
EXAMPLE CONSOLE ACCESS FOR AWS DeepComposer
```

Allow users to view their own permissions

This example shows how you might create a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the console or programmatically using the AWS CLI or AWS API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
```

```
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Troubleshooting AWS DeepComposer AWS Identity and Access Management

Use the following information to help you diagnose and fix common issues that you might encounter when working with DeepComposer and IAM.

Topics

I am not authorized to perform an action in DeepComposer

If the AWS Management Console tells you that you're not authorized to perform an action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a *`widget`* but does not have `DeepComposer:`*`GetWidget`* permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 DeepComposer:GetWidget on resource: my-example-widget
```

In this case, Mateo asks his administrator to update his policies to allow him to access the *`my-example-widget`* resource using the `DeepComposer:`*`GetWidget`* action.

I am not authorized to perform `iam:PassRole`

If you receive an error that you're not authorized to perform the `iam:PassRole` action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password. Ask that person to update your policies to allow you to pass a role to DeepComposer.

Some AWS services allow you to pass an existing role to that service, instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in DeepComposer. However, the action requires the service to have permissions granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform: iam:PassRole
```

In this case, Mary asks her administrator to update her policies to allow her to perform the `iam:PassRole` action.

I want to view my access keys

After you create your IAM user access keys, you can view your access key ID at any time. However, you can't view your secret access key again. If you lose your secret key, you must create a new access key pair.

Access keys consist of two parts: an access key ID (for example, `AKIAIOSFODNN7EXAMPLE`) and a secret access key (for example, `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`). Like a user name and password, you must use both the access key ID and secret access key together to authenticate your requests. Manage your access keys as securely as you do your user name and password.

> **Important**
> Do not provide your access keys to a third party, even to help find your canonical user ID. By doing this, you might give someone permanent access to your account.

When you create an access key pair, you are prompted to save the access key ID and secret access key in a secure location. The secret access key is available only at the time you create it. If you lose your secret access key, you must add new access keys to your IAM user. You can have a maximum of two access keys. If you already have two, you must delete one key pair before creating a new one. To view instructions, see Managing access keys in the *IAM User Guide*.

I'm an administrator and want to allow others to access DeepComposer

To allow others to access DeepComposer, you must create an IAM entity (user or role) for the person or application that needs access. They will use the credentials for that entity to access AWS. You must then attach a policy to the entity that grants them the correct permissions in DeepComposer.

To get started right away, see Creating your first IAM delegated user and group in the *IAM User Guide*.

I want to allow people outside of my AWS account to access my AWS DeepComposer resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether DeepComposer supports these features, see How AWS DeepComposer works with IAM (p. 43).
- To learn how to provide access to your resources across AWS accounts that you own, see Providing access to an IAM user in another AWS account that you own in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see Providing access to AWS accounts owned by third parties in the *IAM User Guide*.
- To learn how to provide access through identity federation, see Providing access to externally authenticated users (identity federation) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see How IAM roles differ from resource-based policies in the *IAM User Guide*.

Data protection in AWS DeepComposer

The AWS shared responsibility model applies to data protection in AWS DeepComposer. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. This content includes the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the Data Privacy FAQ. For information about data protection in Europe, see the AWS Shared Responsibility Model and GDPR blog post on the *AWS Security Blog*.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual user accounts with AWS Identity and Access Management (IAM). That way each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We recommend TLS 1.2 or later.
- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing personal data that is stored in Amazon S3.
- If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-2.

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form fields such as a **Name** field. This includes when you work with AWS DeepComposer or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Data encryption

The following information explains where AWS DeepComposer uses data encryption to protect your data. Encryption is configured by default for your data in AWS DeepComposer.

AWS DeepComposer collects and stores data related to compositions you have uploaded and created in the AWS DeepComposer music studio. AWS DeepComposer also creates and stores data related to models you have trained in the AWS DeepComposer console.

Encryption at rest

AWS DeepComposer works with AWS Key Management Service (AWS KMS) to provide enhanced encryption for your data at rest.

Encryption in transit

Data stored in Amazon S3 and Amazon DynamoDB are encrypted in transit using HTTPS SSL.

Compliance Validation for AWS DeepComposer

AWS DeepComposer is not in scope of any AWS compliance programs.

To learn whether AWS DeepComposer or other AWS services are in scope of specific compliance programs, see AWS Services in Scope by Compliance Program. For general information, see AWS Compliance Programs.

You can download third-party audit reports using AWS Artifact. For more information, see Downloading Reports in AWS Artifact.

Your compliance responsibility when using AWS services is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- Security and Compliance Quick Start Guides – These deployment guides discuss architectural considerations and provide steps for deploying baseline environments on AWS that are security and compliance focused.
- Architecting for HIPAA Security and Compliance Whitepaper – This whitepaper describes how companies can use AWS to create HIPAA-compliant applications.

  Note
  Not all services are compliant with HIPAA.
- AWS Compliance Resources – This collection of workbooks and guides might apply to your industry and location.
- Evaluating Resources with Rules in the *AWS Config Developer Guide* – The AWS Config service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- AWS Security Hub – This AWS service provides a comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices.
- AWS Audit Manager – This AWS service helps you continuously audit your AWS usage to simplify how you manage risk and compliance with regulations and industry standards.

Infrastructure security in AWS DeepComposer

As a managed service, AWS DeepComposer is protected by the AWS global network security procedures that are described in the Amazon Web Services: Overview of Security Processes whitepaper.

You use AWS published API calls to access AWS DeepComposer through the network. Clients must support Transport Layer Security (TLS) 1.0 or later. We recommend TLS 1.2 or later. Clients must also support cipher suites with perfect forward secrecy (PFS) such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.

AWS DeepComposer-dependent AWS services

When you use AWS DeepComposer to create music and train models, AWS DeepComposer uses the following services:

AWS CloudFormation

This service is used to create training jobs for AWS DeepComposer models.

Amazon DynamoDB

This service is used to store metadata related to compositions and models created using AWS DeepComposer.

AWS Lambda

This service is used to create and run inferences in the AWS DeepComposer music studio.

Amazon SageMaker

This service is used to train AWS DeepComposer models and perform inference in the AWS DeepComposer music studio.

Amazon Simple Storage Service (Amazon S3)

This service is used to save AWS DeepComposer compositions and models.

AWS Lambda, AWS CloudFormation, and SageMaker, in turn, use other AWS services including Amazon CloudWatch and Amazon CloudWatch Logs.

If there is a service outage for one or more of these services, AWS DeepComposer may be unavailable. To check for a service outage, see the AWS Service health dashboard.

Browser support for AWS DeepComposer

Browser support for the AWS DeepComposer console varies, depending on the feature that you are accessing. The Web Audio API and WebMIDI API are used to connect the keyboard and play songs in the console respectively.

The following table shows which features are available in different browsers.

Browser	Version	Physical Keyboard Playback	Digital Keyboard Playback	Playing Music
Chrome	Latest 3	**Yes**	**Yes**	**Yes**
Firefox	Latest 3	**No**	**Yes**	**Yes**
Safari	Latest 3	**No**	**Yes**	**Yes**
Internet Explorer	11	**No**	**No**	**No**
Edge	Latest	**Yes**	**Yes**	**Yes**
Opera	Latest	**Yes**	**Yes**	**Yes**

AWS DeepComposer console functions

- **Physical Keyboard Playback** – To record your own musical tracks the WebMIDI API operation is required, which limits some browser playback.
- **Digital Keyboard Playback** – To use the digital keyboard inside the AWS DeepComposer console the WebAudio API operation is required, which limits some browser playback.
- **Playing Music** – To play music inside the console the WebAudio API operation is required, which limits some browser compatibility.

Tagging with AWS DeepComposer

A *tag* is label that you or AWS optionally defines and associates with AWS resources, including certain types of AWS DeepComposer resources and actions.

In the AWS DeepComposer service, you can tag models trained in the AWS DeepComposer console and compositions you have created using AWS DeepComposer music studio.

For more information about using tags, use Tagging best practices.

Managing Tags

A tag key can contain as many as 128 characters. A tag value can contain as many as 256 characters. The characters can be Unicode letters, numbers, white space, or one of the following symbols: _ . : / = + -. The following additional restrictions apply to tags:

- Tag keys and values are case sensitive.
- For each associated resource, each tag key must be unique and have only one value.
- Tags are available only for your AWS account, not any other accounts that share the resource. In addition, the tags are available only for resources that are located in the specified AWS Region for your AWS account.
- Do not use `aws:`, `AWS:` or any upper or lowercase combination of such as a prefix for keys, because it is reserved for AWS use. You cannot edit or delete tag keys with this prefix. Values can have this prefix. If a tag value has `aws` as its prefix but the key does not, then AWS DeepComposer considers it to be a user tag and it counts against the limit of 50 tags. Tags with only the key prefix of `aws` do not count against your tags-per-resource limit.

You can use list below to find more information about creating and managing tags for either models or compositions created in AWS DeepComposer.

- Creating and editing tags for models trained in AWS DeepComposer (p. 54)
- Creating and editing tags for compositions you have created in the AWS DeepComposer music studio. (p. 55)

Using tags in IAM policies

After you start implementing tags, you can apply tag-based, resource-level permissions to AWS Identity and Access Management (IAM) policies. This includes operations that support adding tags to resources when resources are created. By using tags in this way, you can implement granular control of which groups and users in your AWS account have permission to access resources in AWS DeepComposer.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "VisualEditor1",
         "Effect": "Allow",
         "Action": [
            "deepcomposer:GetComposition",
            "deepcomposer:UntagResource"
```

```
            ]
            "Resource": "*",
            "Condition": {
               "StringEquals: {
                  "aws:ResourceTag/tagKey": "tagValue"
               }
            }
         }
      ]
}
```

If you define tag-based, resource-level permissions, the permissions take effect immediately. This means that your resources are more secure as soon as they're created, and you can quickly start enforcing the use of tags for new resources. You can also use resource-level permissions to control which tag keys and values can be associated with new and existing resources. For more information, see Controlling access using tags in the *AWS IAM User Guide*.

Creating and editing tags for models trained in AWS DeepComposer

You can use the procedures below to create and manage tags for models in the AWS DeepComposer console.

Adding a tag while training a model in the AWS DeepComposer console

1. Open the AWS DeepComposer console.
2. In the navigation pane, choose **Models**.
3. On the **Models** page, choose **Create a model**.
4. On the **Train a model** page, under **Generative algorithm**, choose an available option.
5. Choose an available **Training dataset**.
6. (Optional) Under **Hyperparameters**, update the default values to adjust how your custom model is trained.
7. Under **Model details**, give your model a name and an optional description.
8. Under **Tags**, choose **Add a new tag**.
9. For **Key** and **Value**, enter appropriate values.

   For example, **`BachModels`** and **`Model01`**, respectively.
10. (Optional) To add another tag, choose **Add new tag**.
11. (Optional) To remove a tag, choose **Remove**.
12. Choose **Start training** to begin training your model.

After tagging and submitting your new model for training, you can manage its tags during training or after training has completed. Use the procedure below to manage existing tags.

Managing tags for AWS DeepComposer models

1. Open the AWS DeepComposer console.
2. In the navigation pane, choose **Models**.
3. On the **Models** page, choose your model.
4. Choose **View details**.

5. Under **Tags**, choose **Manage tags**.
6. (Optional) To remove an existing tag, choose **Remove**.
7. (Optional) To add a new tag, choose **Add new tag**.
8. For **Key** and **Value**, enter appropriate values.

   For example, **`PopModels`** and **`Model01`**, respectively.
9. When finished managing your tags for that specific model, choose **Save changes**.

Creating and editing tags for compositions created in the AWS DeepComposer music studio

You can create and edit tags for compositions you have created in the AWS DeepComposer music studio. Unlike models, you can only create and edit tags after you have generated the composition in the AWS DeepComposer music studio.

Managing tags for AWS DeepComposer compositions

1. Open the AWS DeepComposer console.
2. In the navigation pane, choose **Compositions**.
3. On the **Compositions** page, choose you composition.
4. In the **Actions** drop down menu, choose **Manage tags**.
5. To add a new tag, choose **Add new tag**.
6. For **Key** and **Value**, enter appropriate values.

   For example, **`JazzModel`** and **`Model01`**, respectively.
7. (Optional) To remove an existing tag, choose **Remove**.
8. When finished managing your tags for that specific composition, choose **Save**.

Additional information

For more information about tagging, see the following resources.

- AWS Tagging Principles in the *AWS General Reference*
- AWS Tagging Strategies (downloadable PDF)
- AWS Access Control in the *AWS IAM User Guide*
- AWS Tagging Policies in the *AWS Organizations User Guide*

Document history for AWS DeepComposer Developer Guide

The following table describes the important changes to the documentation since the initial release of AWS DeepComposer.

update-history-change	update-history-description	update-history-date
Initial release AWS DeepComposer Developer Guide (p. 56)	Initial release of the documentation.	December 2, 2019

AWS glossary

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