https://repost.aws/knowledge-center/bedrock-prompts-deterministic-responses

How do I optimize prompts to achieve deterministic responses on Amazon Bedrock?

I want to use effective prompts and features to get more consistent and reliable results in Amazon Bedrock.

Resolution

If your Amazon Bedrock prompts don't function correctly, or you want better deterministic responses, then use the following solutions.

Select the right foundation model

Amazon Bedrock provides access to multiple pretrained foundation models (FMs), such as Amazon Titan, Anthropic Claude, and Mistral.
To find the right FM for your use case, take the following actions:

- Adjust model inference parameters, such as **temperature** and **top-p**, to reduce randomness in the generated text.
- Test different models. To perform a basic test, use the **Compare mode** feature on the Amazon Bedrock console's **Chat** playground.
- Use different models, and then compare the consistency of responses across multiple runs of the same prompt.
- Choose the model that has the most deterministic behavior for your use case.

Use model evaluation jobs

To perform an advanced test, use model evaluation jobs. To identify potential issues or biases in the prompt or training data, use an automatic model evaluation. To check the output consistency on the input data, use a model evaluation job that uses human workers.

Create clear and specific prompts

To create clear and specific prompts that generate the appropriate response, take the following actions:

- Write the prompt in different ways. For example, change the structure, keywords, or format to get more consistent responses from the model.
- Use prompts that limit the model's ability to generate open-ended or divergent outputs.

- Adjust the FM's inference parameters to check for variations in the outputs for each prompt.

Example of an effective prompt:

```
 ``` You are a professional proofreader. Please review the following text for any grammatical errors, spelling mistakes, or inconsistencies, and provide corrections:

[Insert text to be proofread] ```
```

The preceding prompt clearly defines the model's role and provides specific instructions. It also limits the scope of the response, which increases the chances of deterministic and relevant outputs.

Customize your FM

If the pretrained FMs don't work for your specific requirements, then you can use continued pretraining and fine tuning to customize the FMs.

To fine tune a model, complete the following steps:

1. Prepare a dataset that represents the kinds of inputs that you want to use with the FM.
2. Fine tune or use continued pretraining to train the FM on your dataset. Adjust the custom model hyperparameters as needed.
3. Evaluate the fine-tuned FM's performance, and then repeat the steps until you get the appropriate results. For more information, see Guidelines for model customization.

Combine and refine FM outputs

Ensemble techniques

To get more accurate and reliable results, set up a workflow that combines outputs from multiple models to improve the final prediction's reliability and robustness. You can also add custom logic to analyze and refine the outputs.

To modify or remove non-deterministic elements from the outputs, use post-processing tasks.

Prompt chaining

Use chained prompts to build more sophisticated and capable generative AI applications.

Break down your complex task into smaller, more manageable subtasks that have their own prompt. Then, combine the subtasks to form the complete complex task that you want the FM to accomplish. Use Amazon Bedrock with AWS Step Functions to arrange the prompts in a predefined order or by a defined set of rules.
```

To get started, see the amazon-bedrock-serverless-prompt-chaining repository on the GitHub website. For information about how to construct effective AI applications, see Build and orchestrate generative AI applications with Amazon Bedrock and Step Functions.

Set up guardrails

To control the responses from the FMs, set up guardrails for Amazon Bedrock. You can consistently apply the guardrails across multiple FMs. Use guardrails to block certain topics, filter out harmful or unpredictable content, remove undesirable words, and hide sensitive information. For more information, see Guardrails for Amazon Bedrock.

You can also customize blocked messages and use the built-in test window to test your guardrail configurations. To get controlled and predictable outputs, connect guardrails directly with the FMs when you generate responses.

How do I resolve the error "Failed to receive X resource signal(s) within the specified duration" for EC2 Windows instances in AWS CloudFormation?

I receive the error message: "Failed to receive X resource signal(s) within the specified duration" for Windows Amazon Elastic Compute Cloud (Amazon EC2) instances in AWS CloudFormation.

Short description

You get this error when CloudFormation doesn't receive success signals for resources that have a CreationPolicy attribute specified with a **ResourceSignal** in it. The error might occur for an Amazon EC2 instance, Auto Scaling group, or a wait condition.

Note: The following resolution applies only to CloudFormation stacks that you create with Windows instances. For Linux instances, see How do I resolve the error "Failed to receive X resource signal(s) within the specified duration" in AWS CloudFormation?

Resolution

Based on your use case, use the following troubleshooting steps to resolve your issue.

Note: To prevent a stack rollback, choose **Preserve successfully provisioned resources** for Stack failure options in the CloudFormation console. When you choose this option, the failed instance isn't terminated until you delete the stack.

The cfn-bootstrap MSI package isn't installed on one or more instances of the AWS CloudFormation stack

To confirm that the cfn-signal script is installed on the instance that's configured to send signals to the stack, complete the following steps:

1. Use RDP to connect to your Windows instance.
2. Confirm that the cfn helper scripts package is installed. Run the following command in Windows PowerShell:

```
Get-Package -name aws-cfn-bootstrap
```

Important: By default, CloudFormation helper scripts are installed on Amazon Windows Amazon Machine Images (AMIs). To install the helper scripts, see CloudFormation helper scripts reference.

The AWS CloudFormation template has syntax errors or incorrect values

To find the errors and incorrect values, complete the following steps:

1. In a code editor, open the CloudFormation template for your stack. Then, find the **UserData** property section.
2. Check for errors with syntax, missing spaces, misspellings, and other typos.
3. Confirm that the values for the stack, resource, and AWS Region properties are correct.
   Note: Check for syntax errors or incorrect values in the bootstrap script that's included in the **UserData** property. The script calls the cfn-signal.

If you signal within the **cfn-init** commands, then look for information about the signal in the **cfn-init** logs. To search for errors in the cloud-init or cfn-init logs, use RDP to connect to your instance. Then, use the keyword "error" or "failure" to search for detailed error or failure messages in the following logs:

```
C:\cfn\log\cfn-init.log
C:\cfn\log\cfn-init-cmd.log
C:\cfn\log\cfn-wire.log
```

The timeout property for the CreationPolicy attribute is very low

The **timeout** property's value is defined by the **CreationPolicy** attribute. Confirm that the value is high enough to run tasks before the cfn-signal script sends signals to CloudFormation resources.

To check the **timeout** property value and compare the signaling and resource failure timestamps, complete the following steps:

1. In a code editor, open the CloudFormation template for your stack to find the **timeout** property value.
   Note: The **timeout** property value is the maximum amount of time that CloudFormation waits for a signal before it returns an error.
2. To get an estimate of when the cfn-signal script is activated, use RDP to connect to the instance. Then, run the following command:

   ```
   C:\cfn\log\cfn-init.log
   ```

   Compare the start and end timestamps that are logged in the file to get an estimate of the time it took to bootstrap. Modify the timeout value as needed. The maximum time that you can specify is 12 hours.

The log file shows a timestamp when the **SUCCESS** signal is sent to CloudFormation resources.
Example:

```
2019-01-11 12:46:40,101 [DEBUG] Signaling resource EC2Instance in stack XXXX
with unique ID i-045a536a3dfc8ccad and status SUCCESS
```

3. Open the CloudFormation console.
4. Choose the **Events** view.
5. Choose **Status reason**. Expand the row for the event with the status reason "Failed to receive X resource signal(s) within the specified duration."
6. Compare the signaling timestamp with the resource failure timestamp.
Note: For a successful completion, the script must send the signal before the instance is created or fails to create.

The cfn-signal isn't sent from the Amazon EC2 instance

Verify that the signal that CloudFormation received came from the instance. Check the cfn wire log that's available at **C:\cfn\log\cfn-wire.log**. If the response isn't 200, then there might be a connectivity issue between your instance and the CloudFormation's endpoint.

If you configure a reboot in your instance and the cfn-signal is set in the **UserData** section, then the signal might not be sent. This is because **UserData** runs only once. For more information, see How do I stop my Amazon EC2 Windows instance from signaling back as CREATE_COMPLETE before the instance finishes bootstrapping?

When you send signals from somewhere that's not your instance, use the SignalResource API. For example, you can use an AWS Lambda function to call the SignalResource API, and then send the signal to the stack. If you get an error, then use CloudWatch Logs to check your Lambda logs to understand why the signal wasn't sent to the stack.

How do I use the unified CloudWatch agent to troubleshoot log timestamp issues?

I want to use the unified Amazon CloudWatch agent to troubleshoot log timestamp issues.

Short description

The unified CloudWatch agent uses the PutlogEvents API to upload a batch of log events to Amazon CloudWatch Logs. Log events in a batch can't be more than 2 hours in the future and can't be more than 14 days old. Also, log events can't be from earlier than the retention period of the log group.

If you have log timestamp issues, then you might receive an error message that's similar to one of the following:

- "<timestamp> E! [outputs.cloudwatchlogs] The log entry in (<Log Group Name>/(<Log Stream Name>) with timestamp (<actual log timestamp>) comparing to the current time (<current timestamp> m=+100) is out of accepted time range. Discard the log entry."
- "<timestamp> W! [outputs.cloudwatchlogs] 1 log events for log '<Log Group Name>/(<Log Stream Name>' are expired."

Resolution

To troubleshoot these errors, complete the following steps:

1. Make sure that you use **timestamp_format** in the unified CloudWatch agent configuration file that specifies the timestamp format.
2. (Optional) If necessary, remove the **timestamp_format** from the unified CloudWatch agent configuration file.
3. Restart the unified CloudWatch agent, and then confirm that the current time is used.

Related information

Why isn't the unified CloudWatch agent pushing log events?