Whitepaper

AWS FSx for Lustre

Deploying an Ephemeral Parallel File System for High Performance Computing

AWS FSx for Lustre

Deploying an Ephemeral Parallel File System for High Performance Computing

Building public cloud High Performance Computing (HPC) environments that require fast parallel file system access hasn't always been easy. While it's a relatively simple task to spin up hundreds or even thousands of instances in AWS, if they all have to share the same data - as is generally the case in HPC - standard file serving solutions such as AWS Elastic File System (EFS) don't typically scale to meet the requirement of keeping the compute running at peak utilization.

Performing HPC in public clouds hasn't always been cost-effective versus on-premise solutions since its workloads are so CPU intensive. Large research institutions, for example, do a very good job of keeping their own HPC infrastructure busy around the clock. But if the workloads are ephemeral - that is they come and go for running occasional jobs or for managing peak loads - they should spin up or down as quickly and efficiently as possible to minimize cost. In such cases bursting HPC workloads into public cloud HPC can be very compelling.

> *"It has been my experience that building HPC environments - all of which require fast and parallel file system access - has never been a simple or easy task. Our work with FSx for Lustre finally enables, and more importantly automates, the timely cost-efficient creation and tear down of HPC in the Cloud."*
>
> - Ludovic Francois, CEO, TrackIt

With growing requirements for on-demand HPC computing driven by rapid advances in industries such as biotech and technologies like machine learning, the need for a cost-effective ephemeral high-performance file system that can be shared among large numbers of compute instances has become increasingly obvious for use in the cloud.

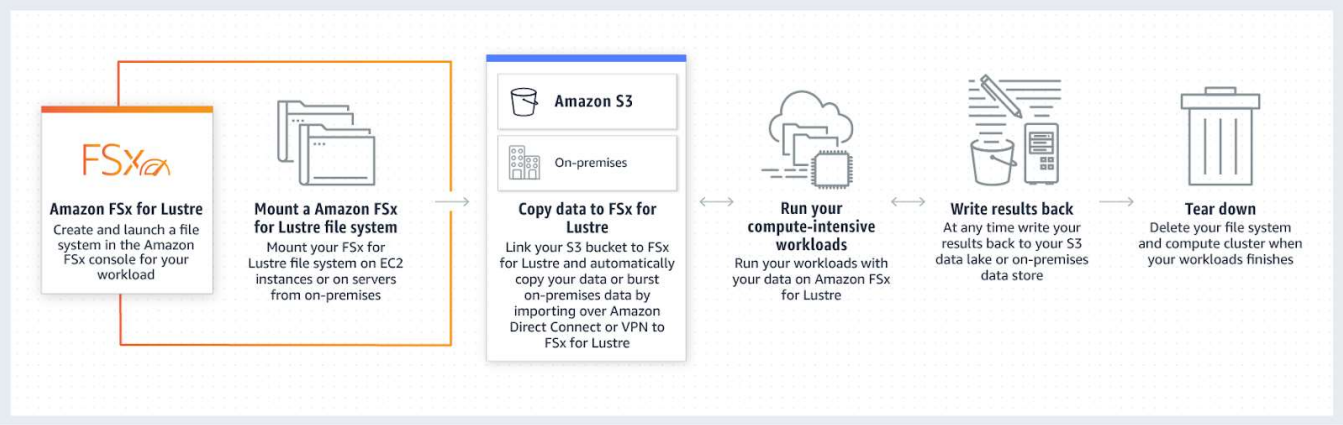
Launched in November 2018, Amazon FSx for Lustre provides a high-performance file system designed for fast processing of workloads. This white paper aims to provide the reader with a framework for building a simple pipeline for the creation of an ephemeral FSx for Lustre file system to be used for compute-intensive environments.

Applicability Across Industries

Biotech	**Media & Entertainment**	**Product Design**
• Genomic Sequencing • Molecular Microscopy	• Animation & VFX Rendering • Processing & Transcoding	• Electronic Design Automation • Mechanical Simulation

Classic Supercomputing	**Emerging Use Cases**
• Fluid Dynamics • Molecular & Particle Simulation • Weather	• Machine Learning • Artificial Intelligence • Internet of Things Analytics

Lifecycle Pipeline for FSx For Lustre



Ingestion

FSx for Lustre requires the source data files to be on S3. If your files are not already there, common data placement options include:

- SFTP
- AWS S3 Transfer Acceleration
- AWS Data Sync
- AWS Storage Gateway

Files are subsequently pushed to a dedicated FSx for Lustre bucket, with an S3 prefix for the current project.

FSx for Lustre Creation

The pipeline for the creation of the FSx for Lustre file system is illustrated in the AWS Design section below.

Notification of Creation to the User's API

During the creation pipeline, different API calls can be made to monitor the creation of the file system. In the pipeline illustrated below, there are API 1 and API 2. Both APIs will be notified once the creation of FSx for Lustre is initiated and then again once the creation of the file system is complete.

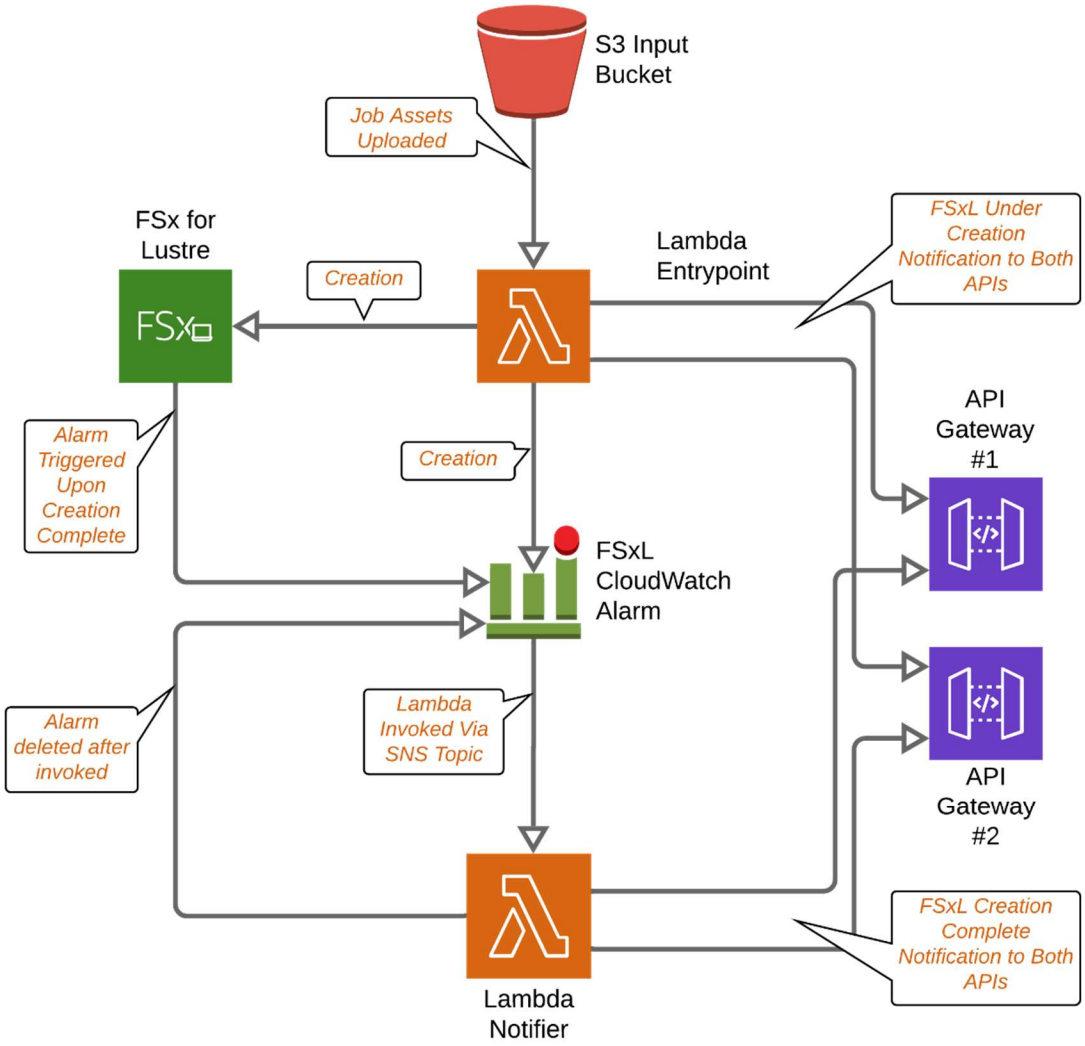
Put event notification

Once all the files are PUT in the S3 bucket, a lambda function is invoked. Lambda functions are small, "serverless" events that can be used to trigger point-in-time actions. This first lambda initiates the creation of the FSx for Lustre file system and also creates a CloudWatch Alarm.

Creation notification

Once the second lambda is invoked by the Cloudwatch Alarm activation, more calls are made to the APIs notifying that the creation of FSx for Lustre is complete.

AWS Design



The design of the pipeline for the creation of the FSx for Lustre is pretty straightforward. The pipeline consists of two main parts:

1. The “entrypoint” lambda: This lambda function is triggered when all the files are uploaded into the input S3 bucket. Its job is to create the FSxL file system along with a CloudWatch alarm that notifies the second lambda once the creation of FSxL is complete.
2. The “notifier” lambda: This lambda function is invoked by the CloudWatch alarm (through a Simple Notification Service (SNS) topic. Its job is to notify the user’s APIs that the creation of the FSx for Lustre file system is complete.

Entrypoint lambda

An S3 event is configured to invoke the lambda once all the files are uploaded. This event includes the bucket name as well as the uploaded object key and is passed as a variable to the lambda function. We use this information as an input for the FSx for Lustre creation. The lambda function then starts the process of creating the FSx for Lustre file system. All the information passed to the creation call is extracted from the S3 event notification. The one piece of information that is not extracted from the S3 event is the SubnetID for which the FSx for Lustre will be created. This information is passed to the lambda via an environment variable configured in Terraform. This lambda function can be configured to make different API requests to register the creation of the FSx for Lustre.

The lambda also creates a Cloudwatch Alarm on the previously created FSx for Lustre. The alarm will watch for the metric *FreeDataStorageCapacity* being greater than 1 on a minimum of 1 datapoint. We don’t really need to know the exact value of *FreeDataStorageCapacity*, but just the fact the metrics are published under this Cloudwatch dimension, as this will let us know that the FSx for Lustre has finished creating. This alarm will, in turn, invoke the second lambda once in an alarm state.
It does not invoke it directly, however.. It’s not possible to invoke a lambda directly from an alarm. The alarm passes the event to an SNS topic, to which the sole subscriber is the second lambda function, which is then invoked whenever an event is published to it.

Notifier lambda

Once an event is sent to the SNS topic from the Cloudwatch alarm, the notifier lambda is invoked. This event contains the Cloudwatch event wrapped in an SNS message structure.
The Cloudwatch event is as follows :


```
{
  "AlarmName": "fsx-creation-watcher-fs-028313d2ae1d969ed",
  "AlarmDescription": null,
  "AWSAccountId": "269868605081",
  "NewStateValue": "ALARM",
  "NewStateReason": "Threshold Crossed: 1 out of the last 1
datapoints [1.210943143936E12 (24/06/19 14:19:00)] was greater
than the threshold (1.0) (minimum 1 datapoint for OK -> ALARM
transition).",
  "StateChangeTime": "2019-06-24T14:20:05.251+0000",
  "Region": "US West (Oregon)",
  "OldStateValue": "INSUFFICIENT_DATA",
  "Trigger": {
    "MetricName": "FreeDataStorageCapacity",
    "Namespace": "AWS/FSx",
    "StatisticType": "Statistic",
    "Statistic": "AVERAGE",
    "Unit": null,
    "Dimensions": [
      {
        "value": "fs-028313d2ae1d969ed",
        "name": "FileSystemId"
      }
    ],
    "Period": 60,
    "EvaluationPeriods": 1,
    "ComparisonOperator": "GreaterThanThreshold",
    "Threshold": 1,
    "TreatMissingData": "",
    "EvaluateLowSampleCountPercentile": ""
  }
}
```


From this event we extract the FileSystemID, which is a Dimension of the metric. This FileSystemID helps describe the associated FSxL and to get its DNS name.
The lambda function then deletes the CloudWatch alarm that triggered it and proceeds to make any additional API calls it has been configured to make.

"Being able to build a true high performance compute environment that is only billed by the second makes The AWS Cloud a compelling and effective way to handle compute jobs that don't necessarily have to run 24/7."

- Brad Winett, President, TrackIt

A Simple Solution That Allows You To Rapidly Deploy Your HPC Workloads

This FSx for Lustre pipeline that we've proposed serves as a push-button solution that allows you to quickly and effectively deploy your high-performance workloads in the Cloud. If you're looking for assistance in building this solution along with customizations specific to your industry, feel free to reach out to us at TrackIt to take advantage of our team's decades worth of experience in building solutions for high-performance computing environments combined with deep AWS expertise.

Open Source Repository: https://github.com/trackit/aws-fsx-for-lustre-ephemeral-storage

References

[1] https://docs.aws.amazon.com/AmazonS3/latest/dev/NotificationHowTo.html
[2] https://docs.aws.amazon.com/lambda/latest/dg/with-s3.html
[3] https://docs.aws.amazon.com/fsx/latest/LustreGuide/fsx-data-repositories.html