PURESTORAGE®

aws partner network | service ready

REFERENCE ARCHITECTURE

Pure FlashBlade for AWS Outposts

Extend the capabilities of AWS Outposts with Pure FlashBlade® to provide fast file and object storage at scale.

Contents

This reference architecture describes how to configure Pure Storage® FlashBlade® as a data source for AWS Outposts. We demonstrate the performance and scalability of this infrastructure solution, including in support of challenging modern workloads like deep learning.

Introduction

AWS Outposts brings the simplicity of the AWS ecosystem to on-premises workloads that have local data processing or low latency requirements. Pure Storage FlashBlade extends the capabilities of AWS Outposts to provide fast file and object storage at scale. FlashBlade delivers a storage system that is simple to deploy, manage, and scale to address your data requirements with AWS Outposts. As organizations utilize larger datasets to solve more business problems, many use cases are arising where data processing needs to occur in near real-time in specific locations to answer questions locally. Cross-location management quickly becomes a burden when each site needs to be managed separately. A simpler solution is to extend the public cloud to private, on-premises locations, building a hybrid-cloud solution.

Key drivers of an AWS Outpost deployment are near–real-time (low latency) processing of data where it originates and processing or aggregation to reduce the amount of data transferred from on-prem to cloud. This reference architecture describes example use cases for the hybrid-cloud solution with Outposts and FlashBlade, provides configuration information, and validates solution performance across file and object protocols.

Hybrid Cloud Storage Requirements

Organizations are finding that, for a variety of reasons, data for some workloads must remain on-premises. These reasons may include security, data governance or regulations, ingress/egress costs, and unreliable or expensive connections from edge to public cloud. When selecting storage for hybrid cloud infrastructure, architects must consider three requirements:

- **Simplicity:** Ease of integration, setup, and deployment to provide a simple and comprehensive hybrid solution. FlashBlade provides a cloud-like experience with Pure1® cloud-based storage management, Evergreen™ architecture, non-disruptive upgrades, and scalability. FlashBlade's complete APIs, mature SDKs, and integration with frameworks like Ansible and Kubernetes simplify managing FlashBlade programmatically as part of regular AWS workflows.
- **Performance:** FlashBlade natively supports high-performance and high-throughput NFS and Amazon S3. FlashBlade supports a range of workload IO profiles, including small files and metadata intensive workloads, and scales as compute scales.

Pure Storage FlashBlade

- **Consolidation:** Enables multiple applications to leverage the same storage platform for all data needs instead of duplicating across silos or using point solutions. FlashBlade provides cloud-optimized efficiency for data and data services.

With this hybrid cloud solution with Outposts and FlashBlade, teams can scale private and hybrid cloud deployments while minimizing management complexity.

About AWS Outposts

AWS Outposts is a fully managed service that extends AWS infrastructure, AWS services, APIs, and tools to virtually any data center, co-location space, or on-premises facility for a truly consistent hybrid experience. AWS Outposts is ideal for workloads that require low latency access to on-premises systems, local data processing, or local data storage.

About Pure Storage FlashBlade

Pure Storage developed the FlashBlade architecture to meet the storage needs of data-driven businesses. FlashBlade is a scale-out all-flash storage system, powered by a distributed file system purpose-built for massive concurrency across all unstructured data types. It can linearly scale capacity and performance simply by adding a single blade at a time, up to 150 blades and 7.8PB capacity. FlashBlade enables consolidation of a range of workloads from backup to analytics and AI onto a single platform. In addition, FlashBlade enables teams to leverage popular cloud consumption and OPEX models through its Pure-as-a-Service™ purchasing model. It can also support CAPEX-based purchasing.

Use Cases for FlashBlade with AWS Outposts

- **High-performance NFS and Amazon S3 Storage for Outposts:** Deliver scalable file and object storage to applications running inside on-prem AWS EC2 instances.
- **AI Workloads:** Deliver low-latency inference at edge locations. Workflows for the development environments in the Region can utilize the same services as production workflows on-premises.
- **Data Hub:** Deliver real-time performance for a range of modern, latency-sensitive applications like Apache Spark, Elasticsearch, AI, and backup/restore concurrently with agile infrastructure.
- **Rapid Recovery:** Deliver Rapid Restore for critical applications and use cases (e.g. recovery from cyberattack) via the same, central backup target that on-prem workloads use. Replicate from FlashBlade to AWS storage in Region seamlessly for cost-effective long-term storage.

System Configuration

This configuration guide describes the steps to enable a pre-installed AWS Outpost to utilize a Pure Storage FlashBlade.

Prerequisites

Configure the Outpost and FlashBlade according to their respective setup guides. AWS provides documentation about how Outposts works, ordering and installing an Outpost, configuring an Outpost, and installing AMIs. For the remainder of this guide, we assume that:

- The Outpost appears as "Active" in the AWS Outposts Console.
- At least one VPC is associated with the Outpost.
- Users can provision EC2 instances within the Outpost.

Architecture Diagram

The architecture of a single-Outpost deployment is described below. As with any Outpost deployment, the two primary layers of the network are the Local Gateway (LGW) network out to the local network and the Service link to the AWS Region. The switches included in the Outpost product ("Outpost devices") connect to the customer's top-of-rack switches ("Customer devices") and then out to the local network and AWS public network.

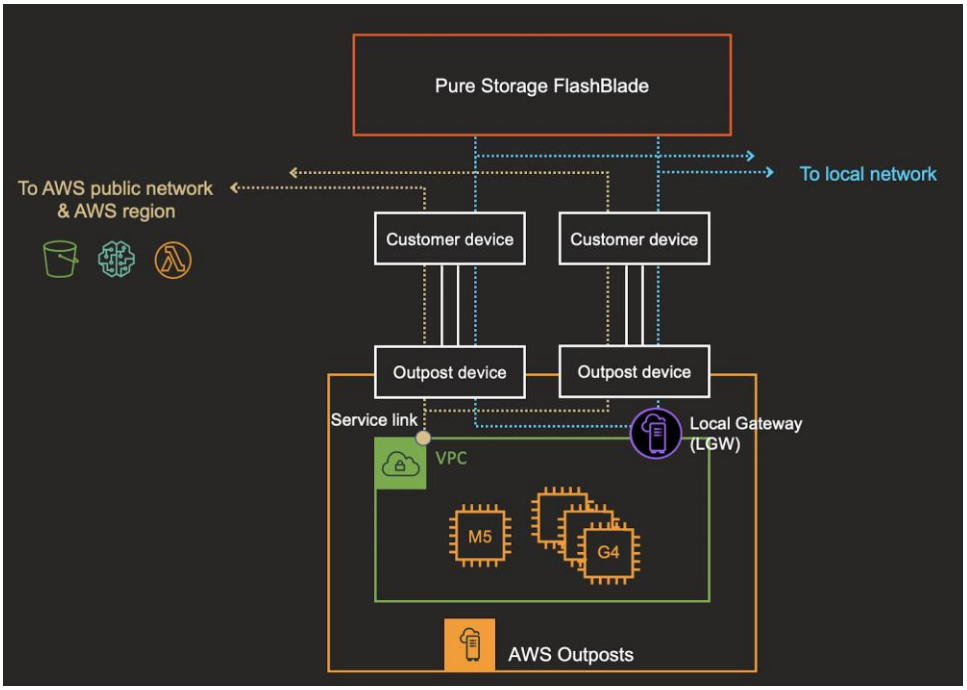


Figure 1. High level topology of the hybrid cloud solution with AWS Outposts and Pure Storage FlashBlade.

Scaling within a Site

As compute demands grow, you can add Outpost compute capacity to an existing Outpost and continue to scale out with additional Outpost racks. Similarly, as storage capacity or performance demands grow, you can add more blades to the FlashBlade system with zero downtime or reconfiguration.

For a comparable look at scaling FlashBlade with on-prem GPUs, see the Hyperscale AIRI documentation for networking best practices for scaling FlashBlade with up to 512 NVIDIA DGX systems.

Multi-site Deployments

Organizations using this hybrid-cloud solution for edge deployments may have multiple edge sites. This solution is built to scale across sites and regions, providing seamless management experience across deployments. You can tune the topology of a multi-site deployment based on the use case or site configuration. Please reach out to AWS or Pure Storage with any questions about planning a multi-site deployment.

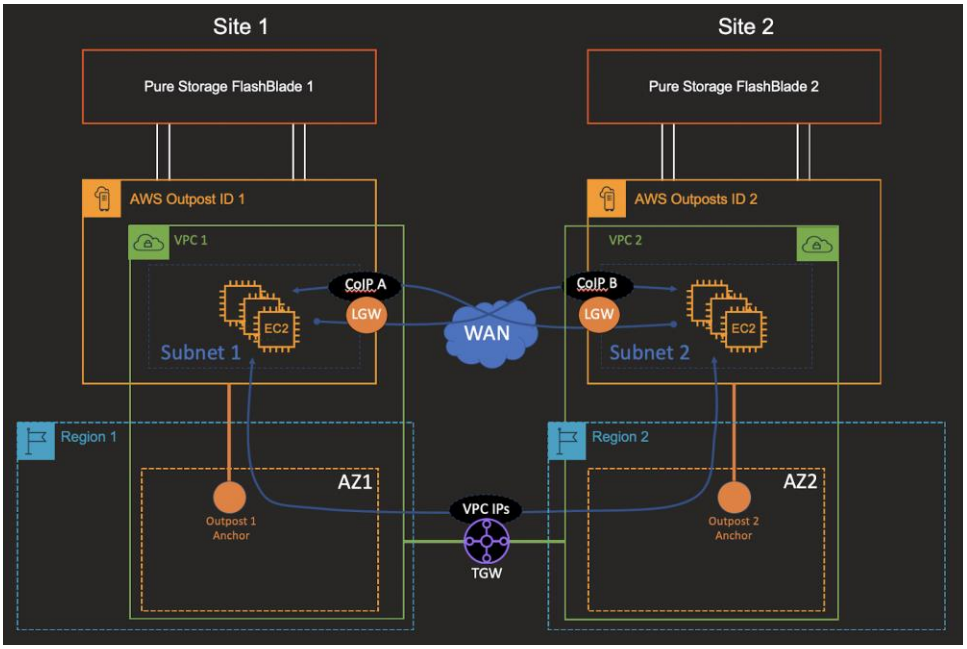
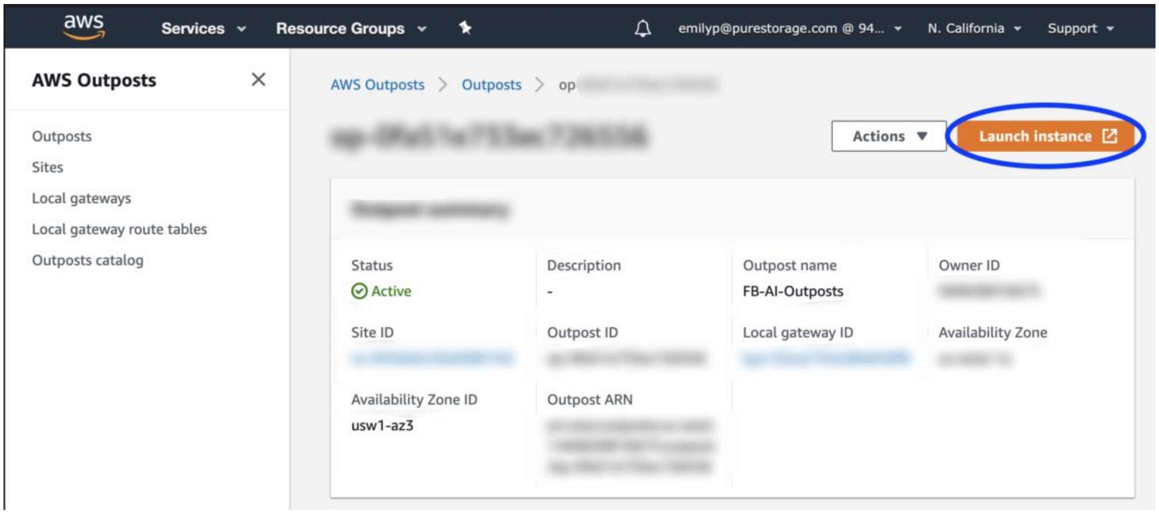


Figure 2. Example multi-site deployment.

Validate Connectivity

Verify that users can launch EC2 instances into the Outpost using the AWS Outposts Console (Figure 3).



[redacted]

[redacted]

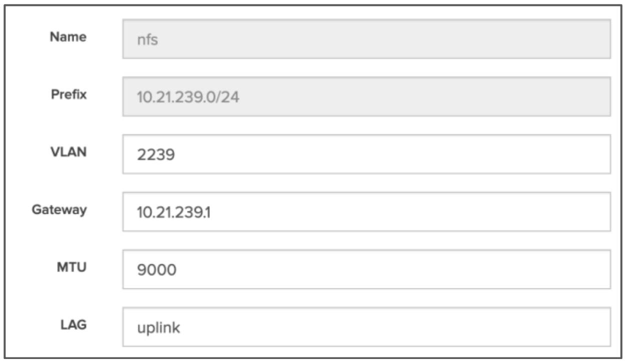
Status: Active
Description: -
Outpost name: FB-AI-Outposts
Owner ID: [redacted]
Site ID: [redacted]
Outpost ID: [redacted]
Local gateway ID: [redacted]
Availability Zone: [redacted]
Availability Zone ID: usw1-az3
Outpost ARN: [redacted]

Figure 3. Use the AWS Outposts console to verify the ability for users to launch EC2 instances:

After launching multiple instances in the Outpost, verify network connectivity between all instances using Ping. Verify that an instance in the Outpost can ping the FlashBlade system.

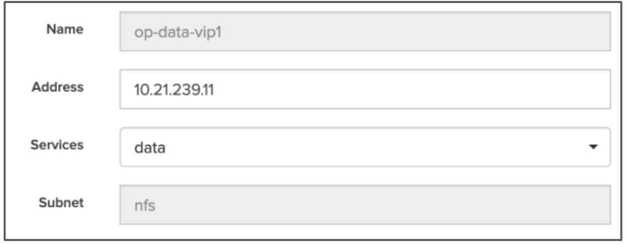
FlashBlade Configuration—NFS

On the Settings > Network tab of the FlashBlade UI, or using the corresponding CLI commands, create a subnet (Figure 4 and 5).



Name	nfs
Prefix	10.21.239.0/24
VLAN	2239
Gateway	10.21.239.1
MTU	9000
LAG	uplink

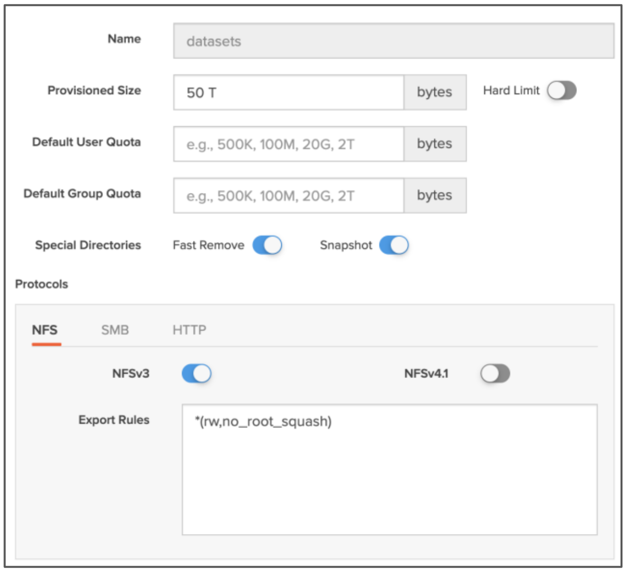
Figure 4. Create a subnet.



Name	op-data-vip1
Address	10.21.239.11
Services	data
Subnet	nfs

Figure 5.

From the Storage > File Systems tab, or using the corresponding CLI commands, create a filesystem with NFS enabled to use for storing datasets (Figure 6).



Name	datasets		
Provisioned Size	50 T	bytes	Hard Limit ○
Default User Quota	e.g., 500K, 100M, 20G, 2T	bytes	
Default Group Quota	e.g., 500K, 100M, 20G, 2T	bytes	
Special Directories	Fast Remove ●	Snapshot ●	

Protocols

NFS SMB HTTP

NFSv3 ● NFSv4.1 ○

Export Rules: *(rw,no_root_squash)

Figure 6. Enable NFS for the new filesystem.

FlashBlade Configuration—Object Store

From the Storage > Object Store tab, create an account. From that account, create a user and select Create Access Key. Like with AWS S3, this key pair is required to access data for the account. Next, create a bucket.

Connecting to FlashBlade File Storage

Mount FlashBlade from within an Amazon EC2 Instance

Use the data vip created previously to mount the filesystem to a directory inside the instance. For example:

```
mkdir -p /mnt/datasets
mount -t nfs 10.21.239.11:/datasets /mnt/datasets
```

Confirm via `ls` that contents of the mounted filesystem appear as expected from within the EC2 instance.

NFS Mount Option: nconnect

Traditionally, an NFS client is bottlenecked by the throughput limits that a single TCP connection can provide, such as the maximum bandwidth from a single NIC on the host to a single blade on the storage server.

To take advantage of the full potential of the FlashBlade and utilize more than one blade to serve traffic to a single host, we utilized the nconnect mount functionality so each instance could use multiple TCP connections (up to 16) to FlashBlade. Thus, in a testing environment where the FlashBlade contains 30 blades, the maximum total read bandwidth across the system is about 30 GB/s. Here's an example command for using the nconnect setting during mount:

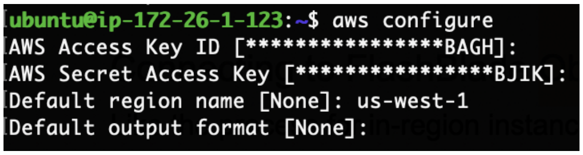
```
mount -t nfs -o ro,nconnect=16 10.21.239.11:/datasets /mnt/datasets
```

Connecting to FlashBlade Object Storage

Like the process for in-Region instances, an IAM role with Amazon S3 access must be associated with the instance. You can find instructions from here: AWS Command Line Interface User Guide.

Browse and Download Data from a FlashBlade Bucket from within an Amazon EC2 Instance

Use the `aws configure` command to add the Access Key ID and Secret Access Key for the FlashBlade object storage user. Also, select your default Region during this step.



```
[ubuntu@ip-172-26-1-123:~$ aws configure
[AWS Access Key ID [****************BAGH]:
[AWS Secret Access Key [****************BJIK]:
[Default region name [None]: us-west-1
[Default output format [None]:
```

Use the `aws s3 ls` command to verify that the EC2 instance has access to buckets on the FlashBlade. Because the system is using a custom endpoint instead of the default AWS S3, specify --endpoint-url with the FlashBlade previously created data VIP.

```
aws --endpoint-url http://10.21.239.11  s3 ls
```

At this point, the system is ready for use.

Support Description

FlashBlade is built for high availability by default. Since there is no single point of failure within a FlashBlade, it requires no further setup or tuning to provide high availability (HA) to workloads.

The Pure Storage Support team provides continuous monitoring and proactive resolution of issues via Pure1 Meta™, our AI-driven storage management platform. Pure provides a best-in-class level of support with a market-leading NPS score of 86 and responses within minutes, not hours or days.

Pure Storage products in general can be sold to any country where permitted by US law.

For assistance setting up FlashBlade with AWS Outposts, please open a case with Pure Storage Support by visiting http://support.purestorage.com or emailing support@purestorage.com.

Test Methodology

The infrastructure used during testing was configured as follows:

Component	Configuration	Networking
AWS Outpost	EC2 Capacity: 4 g4dn.12xlarge 1 m5.24xlarge	8 x 100GbE from Outpost rack out to local network
Pure Storage FlashBlade	7 x 17TB blades (66TB usable, before data reduction)	4 x 40GbE uplinks

Outpost users can deploy any public AWS Machine Image. We tested with AWS Deep Learning AMI (Ubuntu 18.04) V32.0.

Note: This test system used a 1 GbE/s WAN connection back to the AWS Region for management. For workloads that require heavy data transfer between on-premises to AWS regions, users can utilize AWS Direct Connect. More info: Outpost Connectivity to AWS Regions.

Often, latency is a critical metric for applications at edge locations that need to deliver results quickly to edge applications. To validate the performance of FlashBlade as an on-prem storage target for Outposts, we tested NFS latency via fio and object-storage latency via s3-benchmark. NFS latency is measured as "submission latency" and object storage latency is measured as time to first byte.

System Performance

In this section, we validate that FlashBlade can provide low latency data to Amazon EC2 instances running inside an Outpost. Latency can be a key driver for organizations selecting an Outpost. Many business-critical applications can't afford to wait for the time it takes to send data to the public cloud for analysis and receive the response back. When an Outpost is utilized on-premises, it can bring low latency compute to applications. So, it's critical that local storage enables low latency and that queries to storage do not substantially delay the applications.

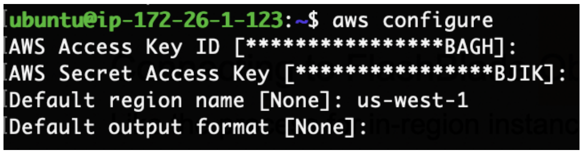
fio command information

```
fio --rw=write --direct=1 --ioengine=posixaio --iodepth=100 --create_serialize=0 --fallocate=none --
group_reporting=1 --startdelay=5 --ramp_time=3 --runtime=600 --time_based=1 --invalidate=1 --blocksize=1024k --
size=4194304k --numjobs=120 --directory=/mnt/datasets create.fio
```

```
fio --rw=read --direct=1 --ioengine=posixaio --iodepth=100 --create_serialize=0 --fallocate=none --
group_reporting=1 --startdelay=5 --ramp_time=3 --runtime=600 --time_based=1 --invalidate=1 --blocksize=1024k --
size=4194304k --numjobs=120 --directory=/mnt/datasets create.fio
```

s3-benchmark command information

Run the `aws configure` command inside an instance to provide the Access Key ID and Secret Access Key for the FlashBlade S3 user.



```
[ubuntu@ip-172-26-1-123:~$ aws configure
[AWS Access Key ID [****************BAGH]:
[AWS Secret Access Key [****************BJIK]:
[Default region name [None]: us-west-1
[Default output format [None]:
```

Execute the benchmark script, using the `–endpoint` option to target FlashBlade as the storage device.

```
./s3-benchmark -endpoint=http://10.21.239.11:80
```

The s3-benchmark test suite will run through a set of object sizes, producing latency results for each. For example, these are results for the smallest and largest object sizes used during the Amazon S3 download test:

```
Download performance with 1 KB objects (m5.12xlarge)
                           +-------------------------------------------------+-------------------------------------------------+
                           |             Time to First Byte (ms)             |             Time to Last Byte (ms)              |
+---------+----------------+-------------------------------------------------+-------------------------------------------------+
| Threads |     Throughput |   avg   min   p25   p50   p75   p90   p99   max |   avg   min   p25   p50   p75   p90   p99   max |
+---------+----------------+-------------------------------------------------+-------------------------------------------------+
|       8 |       6.0 MB/s |     1     1     1     1     1     2     2     3 |     1     1     1     1     1     2     2     3 |
|       9 |       6.9 MB/s |     1     1     1     1     1     1     2     2 |     1     1     1     1     1     2     2     2 |
|      10 |       6.9 MB/s |     1     1     1     1     1     2     4     4 |     1     1     1     1     1     2     4     4 |
|      11 |       7.7 MB/s |     1     1     1     1     1     2     2     4 |     1     1     1     1     2     2     2     4 |
|      12 |       8.3 MB/s |     1     1     1     1     2     2     2     3 |     1     1     1     1     2     2     2     3 |
|      13 |       8.9 MB/s |     1     1     1     1     2     2     2     3 |     1     1     1     1     2     2     2     3 |
|      14 |       8.2 MB/s |     2     1     1     2     2     2     3     5 |     2     1     1     2     2     2     3     5 |
|      15 |       9.2 MB/s |     2     1     1     1     2     2     3     3 |     2     1     1     1     2     2     3     3 |
|      16 |       9.1 MB/s |     2     1     1     2     2     2     3     7 |     2     1     1     2     2     2     3     7 |
+---------+----------------+-------------------------------------------------+-------------------------------------------------+
```

```
Download performance with 512 KB objects (m5.12xlarge)
                           +-------------------------------------------------+-------------------------------------------------+
                           |             Time to First Byte (ms)             |             Time to Last Byte (ms)              |
+---------+----------------+-------------------------------------------------+-------------------------------------------------+
| Threads |     Throughput |   avg   min   p25   p50   p75   p90   p99   max |   avg   min   p25   p50   p75   p90   p99   max |
+---------+----------------+-------------------------------------------------+-------------------------------------------------+
|       8 |     917.5 MB/s |     3     2     2     3     3     3     4     5 |     4     4     4     4     4     5     6     7 |
|       9 |     939.1 MB/s |     3     2     3     3     3     4     6     7 |     5     3     4     5     5     6     8     9 |
|      10 |    1127.0 MB/s |     3     2     2     3     3     3     5     7 |     4     3     4     4     5     5     7     8 |
|      11 |    1228.7 MB/s |     3     2     3     3     3     3     5     6 |     4     3     4     4     5     5     7     9 |
|      12 |    1195.5 MB/s |     3     2     3     3     3     4     5     8 |     5     4     4     5     5     6     8    10 |
|      13 |    1310.8 MB/s |     3     2     3     3     3     4     6     6 |     5     4     4     5     5     6     8     9 |
|      14 |    1253.1 MB/s |     3     2     3     3     4     4     6     6 |     5     4     4     5     6     7    14    17 |
|      15 |    1331.3 MB/s |     4     2     3     3     4     5     7     9 |     6     4     5     5     6     7     9    12 |
|      16 |    1353.5 MB/s |     3     2     3     3     4     4     6     9 |     6     4     5     5     6     8    12    14 |
+---------+----------------+-------------------------------------------------+-------------------------------------------------+
```

The combined latency results are below.

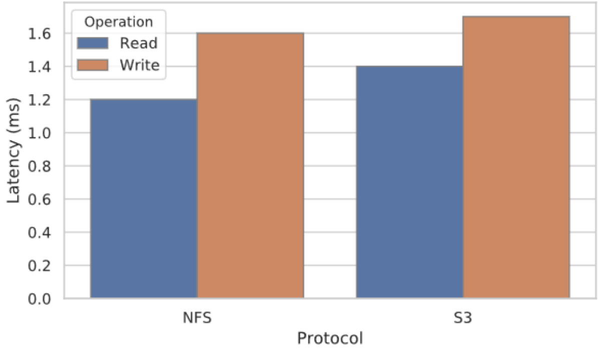


Figure 7. FlashBlade provides low latency for file and object workloads running in EC2 instances on Outposts.

At low load, latency is 1.4ms on average. We also tested the system with higher load and higher concurrent access by adding instances to the workload. Latency remained around 1.4ms. At 75% of the peak throughput, FlashBlade delivers 1.4ms sustained read latency.

For both file and object storage, FlashBlade can provide consistently low latency to applications that require it.

Hybrid AI Workloads

Hybrid-cloud deployments are becoming increasingly common as organizations seek the balance between application performance and management simplicity. In this section, we use example AI workloads to show how this infrastructure solution solves the challenges of hybrid-cloud use cases.

Moving Analysis Closer to Data

One of the primary AI use cases for this hybrid solution is inference at the edge. As organizations publish more models to production, their inference load for deployed models increases, which means they have to support larger-scale data ingest and data processing.

Often the cost or performance penalty is prohibitive to transferring large datasets into the public cloud. Or, the connection from edge to the public cloud may be unreliable. Because of these limitations, organizations with massive data ingest are relocating inference from the public cloud to edge data centers.

Note: in this case, we use "edge" to mean the distributed computing model where service delivery is performed at multiple edge sites, such as colocation facilities.

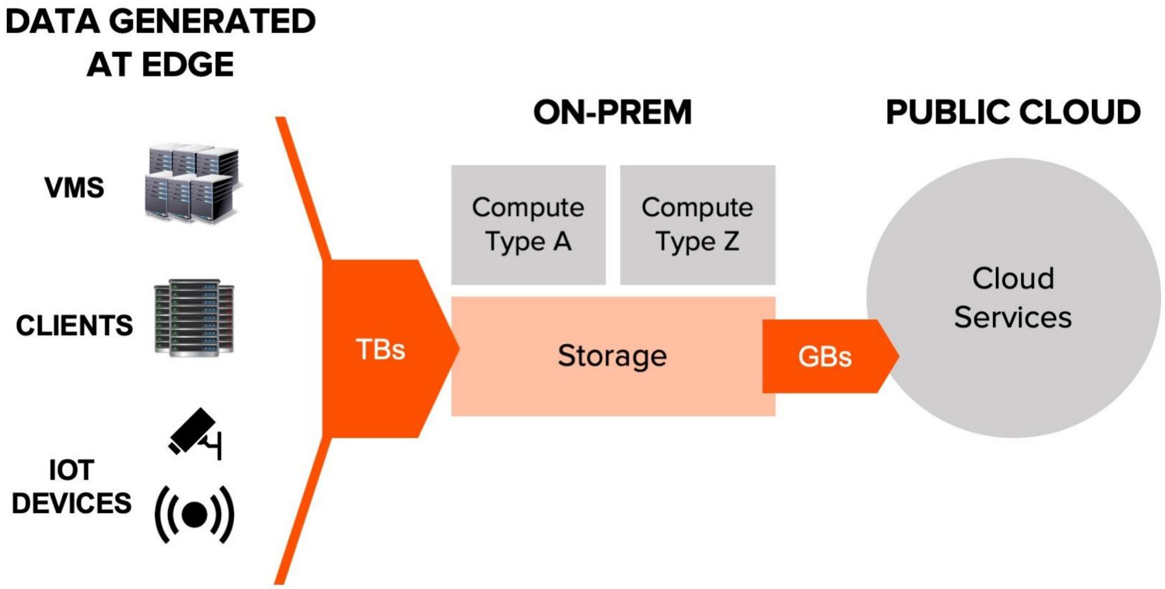


Figure 8. Example hybrid cloud deployment.

In this case, data scientists may perform exploration and model development in the public cloud, and then their models will be deployed into edge data centers for inference. By using AWS Outposts as the compute for their on-prem deployments, users can enjoy the same experience, services, and API across training and inference environments—across development and production environments. And by pairing the Outpost with a FlashBlade as a highly performant, centralized data hub, the data ingest and latency issues are mitigated.

With this deployment model, organizations can easily manage multiple edge locations with custom, regional models deployed, and developers can conduct centralized training—or retraining—of models.

Example Inference Pipeline

Beyond the inference component itself, organizations may also want to incorporate processing components like Amazon EMR / Apache Spark on the Outpost. Often, it’s insufficient to only perform inference directly from raw data at the edge. Many frequently chose to preprocess the data or otherwise manipulate it before running inference. These additional workflows have different compute and storage requirements. The hybrid-cloud solution should be flexible enough to support various pieces of an end-to-end pipeline while maintaining usability and manageability.

For example, in an edge inference pipeline, data may again be ingested from edge IoT devices and edge applications onto the FlashBlade, often using a message-streaming framework like Kafka that requires a persistent storage layer to preserve recent history and ensure delivery. That data is then passed to tools like Spark where it's cleaned, transformed, and stored for filtering analysis. You can use services like AWS EMR to manage those preprocessing steps with the same in-Region experience developers are used to. Finally, in this example, the Spark datasets created onto FlashBlade can be read into Outpost instances for inference.

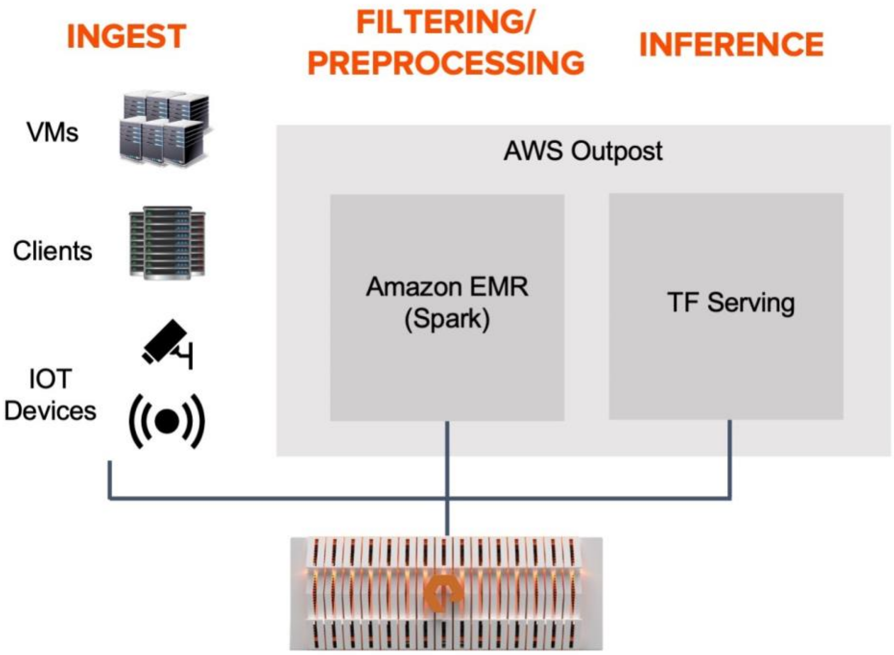


Figure 9. Example on-prem inference pipeline. Both file and object protocol workloads are supported.

The infrastructure is flexible to support this pipeline. For compute, applications can utilize their appropriate instance types within the Outpost. For storage, FlashBlade provides excellent performance across a range of IO types and file sizes, so no tuning is required when users add applications.

Organizations can also use a container layer to help scale, manage, backup, and recover these services. The hybrid-cloud solution with FlashBlade and Outposts can seamlessly support a diverse collection of workloads with low latency and high throughput.

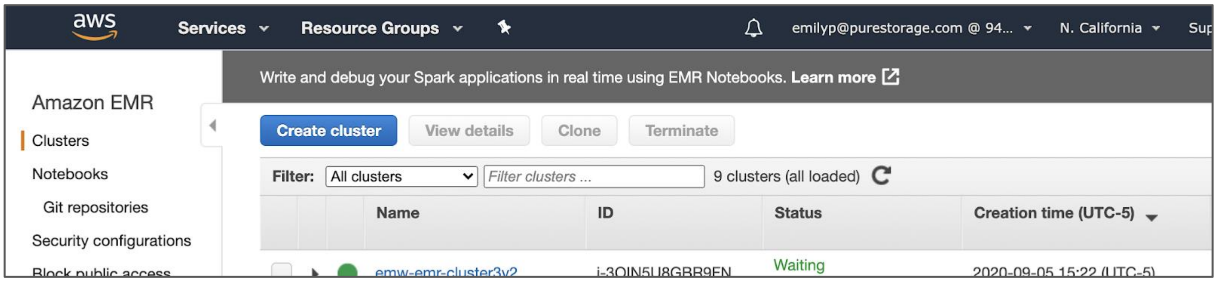
Data architects may choose from a number of different streaming and analysis tools in an inference pipeline like this one. The full configuration details for Apache Kafka or TensorFlow Serving are beyond the scope of this paper as they are not unique to AWS Outposts. Instead, we focus on the configuration of an AWS-curated Elastic MapReduce (EMR) cluster running in an Outposts system with FlashBlade as the local, shared storage.

Configuring EMR Clusters

Use the following steps to configure EMR clusters to run in Outposts and read from FlashBlade S3.

Step 1. Launch the EMR Cluster into Your Outpost

From the EMR Console, in the AWS Region that serves as “home base” for the Outpost, select Create cluster (Figure 10).



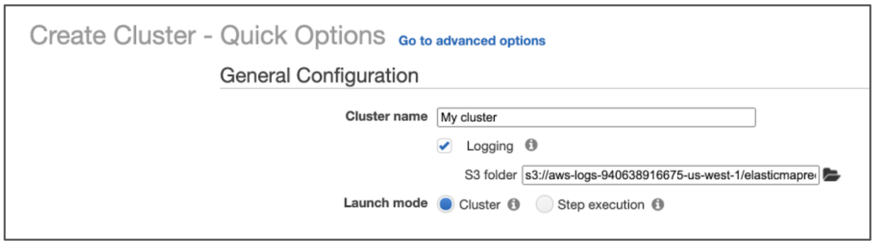
Write and debug your Spark applications in real time using EMR Notebooks. **Learn more**

Filter: All clusters | Filter clusters ... | 9 clusters (all loaded)

Name	ID	Status	Creation time (UTC-5)
emw-emr-cluster3v2	j-3QIN5U8GBB9EN	Waiting	2020-09-05 15:22 (UTC-5)

Figure 10.

The cluster's target location can be switched from in-Region (default) to an Outpost by specifying the Outpost's subnet during cluster creation (Figure 11). Select Go to advanced options to expose more networking options (Figure 12).


Create Cluster - Quick Options   **Go to advanced options**

General Configuration

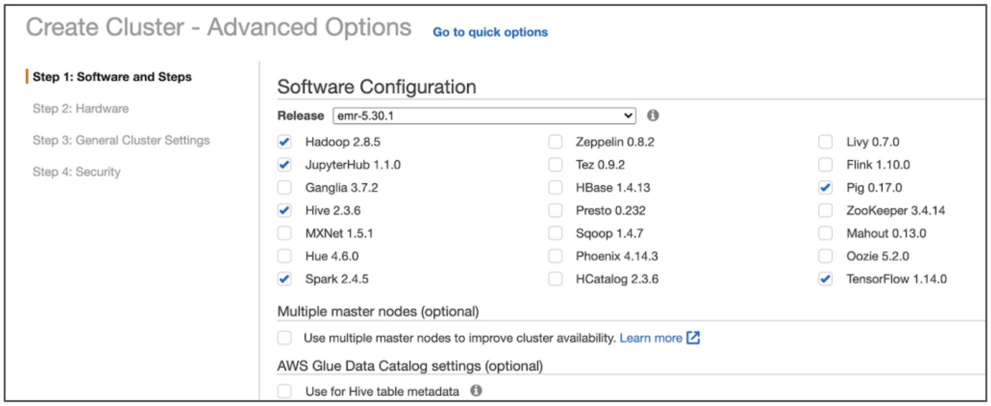
Cluster name: My cluster

☑ Logging

S3 folder: s3://aws-logs-940638916675-us-west-1/elasticmapre

Launch mode: ● Cluster ○ Step execution

Figure 11.


Create Cluster - Advanced Options   **Go to quick options**

Software Configuration

Release: emr-5.30.1

☑ Hadoop 2.8.5
☑ JupyterHub 1.1.0
□ Ganglia 3.7.2
☑ Hive 2.3.6
□ MXNet 1.5.1
□ Hue 4.6.0
☑ Spark 2.4.5
□ Zeppelin 0.8.2
□ Tez 0.9.2
□ HBase 1.4.13
□ Presto 0.232
□ Sqoop 1.4.7
□ Phoenix 4.14.3
□ HCatalog 2.3.6
□ Livy 0.7.0
□ Flink 1.10.0
☑ Pig 0.17.0
□ ZooKeeper 3.4.14
□ Mahout 0.13.0
□ Oozie 5.2.0
☑ TensorFlow 1.14.0

Multiple master nodes (optional)

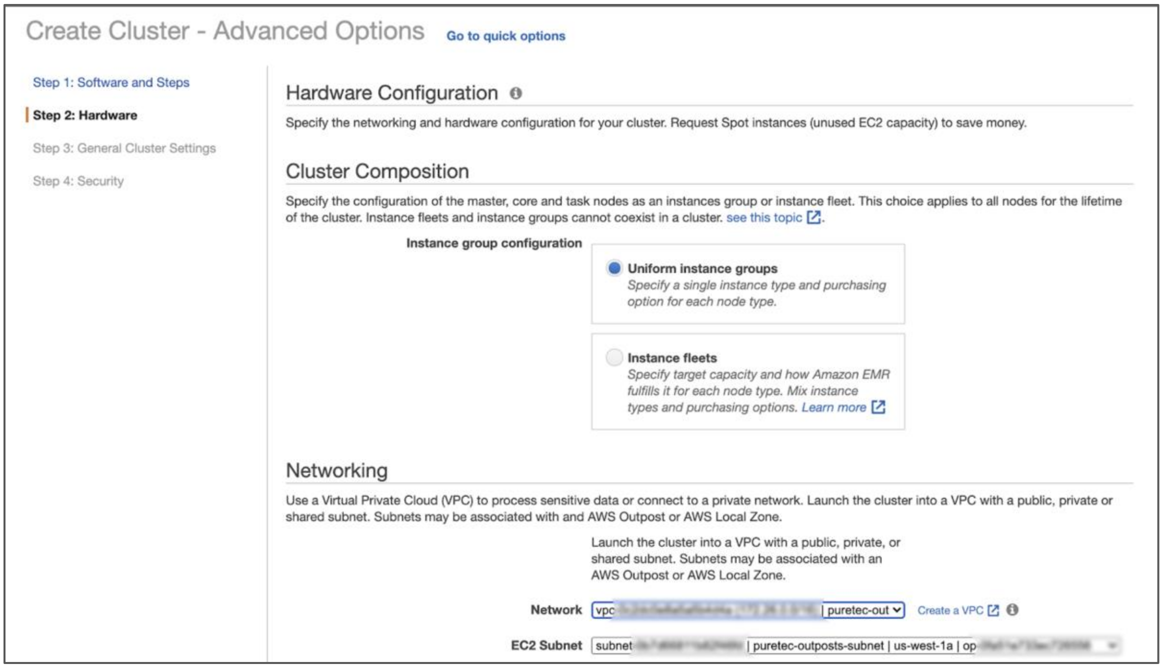
□ Use multiple master nodes to improve cluster availability. Learn more

AWS Glue Data Catalog settings (optional)

□ Use for Hive table metadata

Figure 12.

On the Step 2: Hardware tab, in the Networking section, choose the Outpost's Virtual Private Cloud (VPC). The VPC must already exist and must have a subnet for the Outpost. The EC2 Subnet field should automatically populate with the Outpost's subnet (Figure 13).



Create Cluster - Advanced Options   Go to quick options

- Step 1: Software and Steps
- **Step 2: Hardware**
- Step 3: General Cluster Settings
- Step 4: Security

Hardware Configuration

Specify the networking and hardware configuration for your cluster. Request Spot instances (unused EC2 capacity) to save money.

Cluster Composition

Specify the configuration of the master, core and task nodes as an instances group or instance fleet. This choice applies to all nodes for the lifetime of the cluster. Instance fleets and instance groups cannot coexist in a cluster. see this topic.

Instance group configuration

● **Uniform instance groups**
Specify a single instance type and purchasing option for each node type.

○ **Instance fleets**
Specify target capacity and how Amazon EMR fulfills it for each node type. Mix instance types and purchasing options. Learn more

Networking

Use a Virtual Private Cloud (VPC) to process sensitive data or connect to a private network. Launch the cluster into a VPC with a public, private or shared subnet. Subnets may be associated with and AWS Outpost or AWS Local Zone.

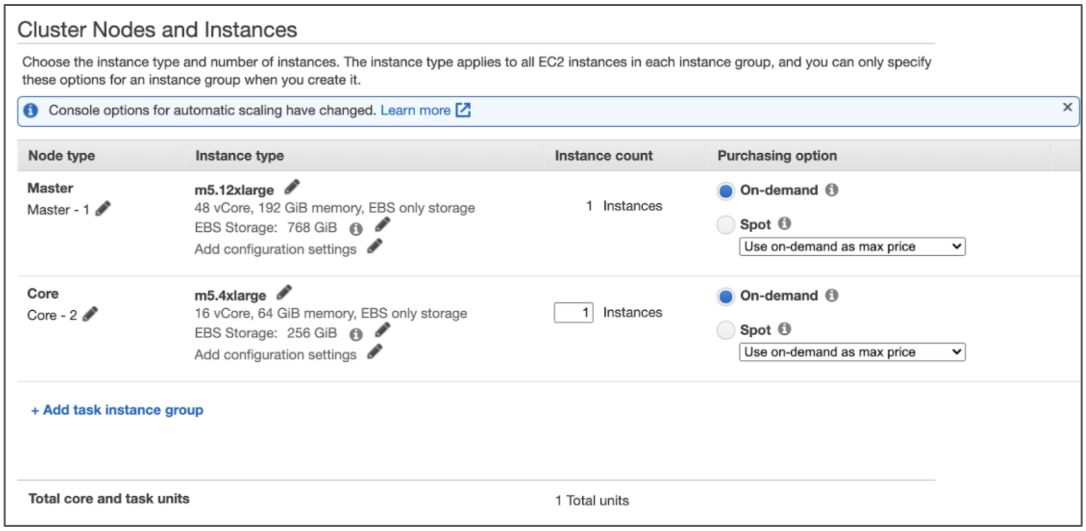
Launch the cluster into a VPC with a public, private, or shared subnet. Subnets may be associated with an AWS Outpost or AWS Local Zone.

Network vpc-[redacted] | puretec-out   Create a VPC

EC2 Subnet subnet-[redacted] | puretec-outposts-subnet | us-west-1a | op-[redacted]

Figure 13.

In the Cluster Nodes and Instances section (Figure 14), be sure to select Instance type and Instance count values that match the instances present within your Outpost.



Cluster Nodes and Instances

Choose the instance type and number of instances. The instance type applies to all EC2 instances in each instance group, and you can only specify these options for an instance group when you create it.

Console options for automatic scaling have changed. Learn more

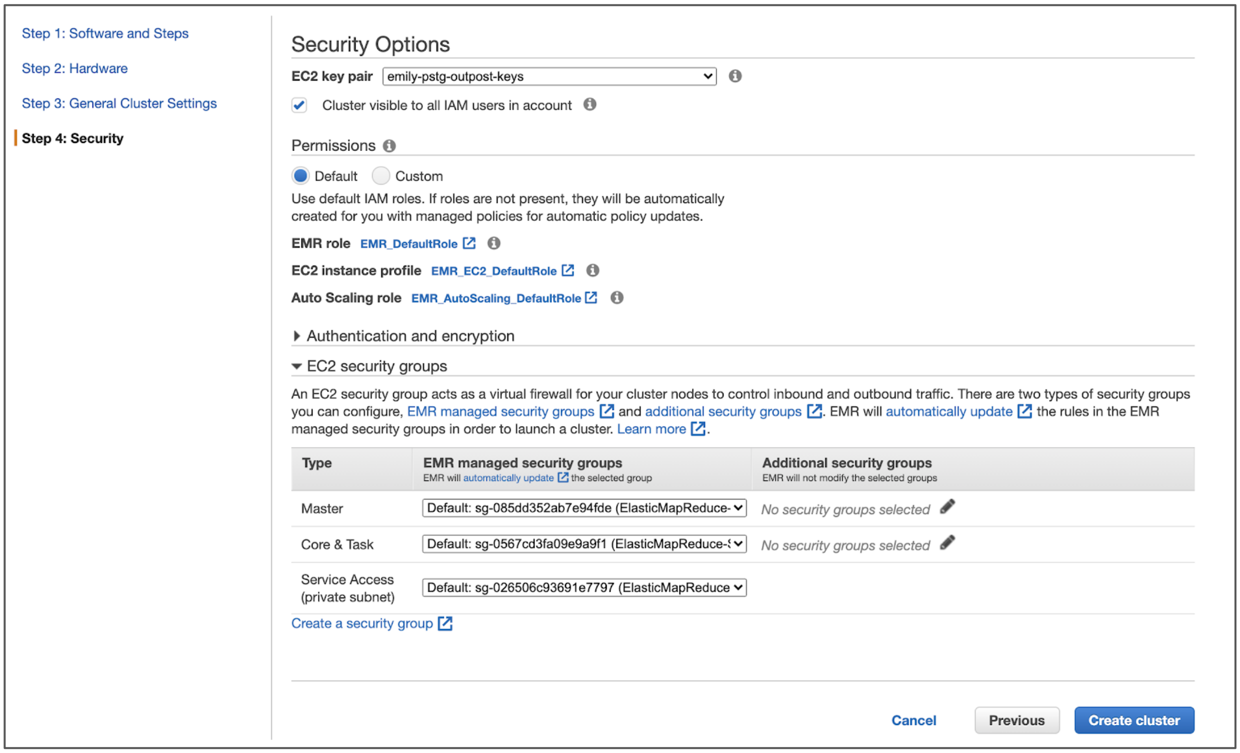
Node type	Instance type	Instance count	Purchasing option
Master Master - 1	**m5.12xlarge** 48 vCore, 192 GiB memory, EBS only storage EBS Storage: 768 GiB Add configuration settings	1 Instances	● **On-demand** ○ **Spot** Use on-demand as max price
Core Core - 2	**m5.4xlarge** 16 vCore, 64 GiB memory, EBS only storage EBS Storage: 256 GiB Add configuration settings	1 Instances	● **On-demand** ○ **Spot** Use on-demand as max price

+ Add task instance group

Total core and task units   1 Total units

Figure 14.

Complete the cluster definition as normal. Make sure to select an EC2 key pair (default value is to proceed without a key pair). When everything is ready, click Create cluster (Figure 15).



Security Options

EC2 key pair emily-pstg-outpost-keys

☑ Cluster visible to all IAM users in account

Permissions

● Default ○ Custom

Use default IAM roles. If roles are not present, they will be automatically created for you with managed policies for automatic policy updates.

EMR role EMR_DefaultRole

EC2 instance profile EMR_EC2_DefaultRole

Auto Scaling role EMR_AutoScaling_DefaultRole

▾ EC2 security groups

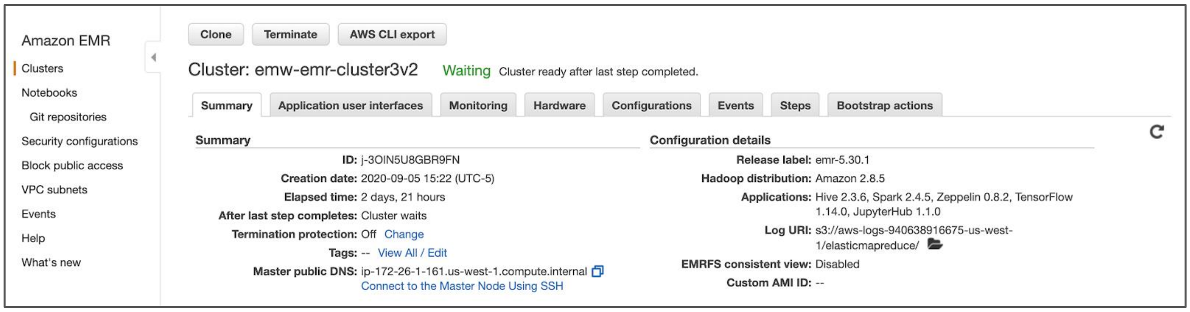
An EC2 security group acts as a virtual firewall for your cluster nodes to control inbound and outbound traffic. There are two types of security groups you can configure, EMR managed security groups and additional security groups. EMR will automatically update the rules in the EMR managed security groups in order to launch a cluster. Learn more.

Type	EMR managed security groups EMR will automatically update the selected group	Additional security groups EMR will not modify the selected groups
Master	Default: sg-085dd352ab7e94fde (ElasticMapReduce-	*No security groups selected*
Core & Task	Default: sg-0567cd3fa09e9a9f1 (ElasticMapReduce-!	*No security groups selected*
Service Access (private subnet)	Default: sg-026506c93691e7797 (ElasticMapReduce	

Figure 15.

Step 2. Give the Cluster Access to FlashBlade S3

From the EMR Console, select the new cluster and **View details** to get to its summary page (Figure 16).



Cluster: emw-emr-cluster3v2 Waiting Cluster ready after last step completed.

Summary

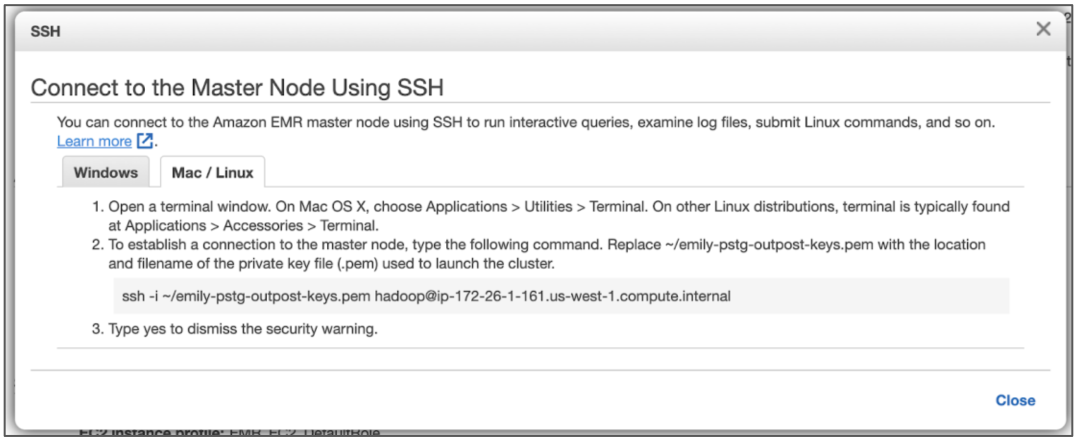
ID: j-3OIN5U8GBR9FN
Creation date: 2020-09-05 15:22 (UTC-5)
Elapsed time: 2 days, 21 hours
After last step completes: Cluster waits
Termination protection: Off Change
Tags: -- View All / Edit
Master public DNS: ip-172-26-1-161.us-west-1.compute.internal
Connect to the Master Node Using SSH

Configuration details

Release label: emr-5.30.1
Hadoop distribution: Amazon 2.8.5
Applications: Hive 2.3.6, Spark 2.4.5, Zeppelin 0.8.2, TensorFlow 1.14.0, JupyterHub 1.1.0
Log URI: s3://aws-logs-940638916675-us-west-1/elasticmapreduce/
EMRFS consistent view: Disabled
Custom AMI ID: --

Figure 16.

Collect SSH login info by clicking **Connect to the Master Node Using SSH** (Figure 17).



Connect to the Master Node Using SSH

You can connect to the Amazon EMR master node using SSH to run interactive queries, examine log files, submit Linux commands, and so on. Learn more.

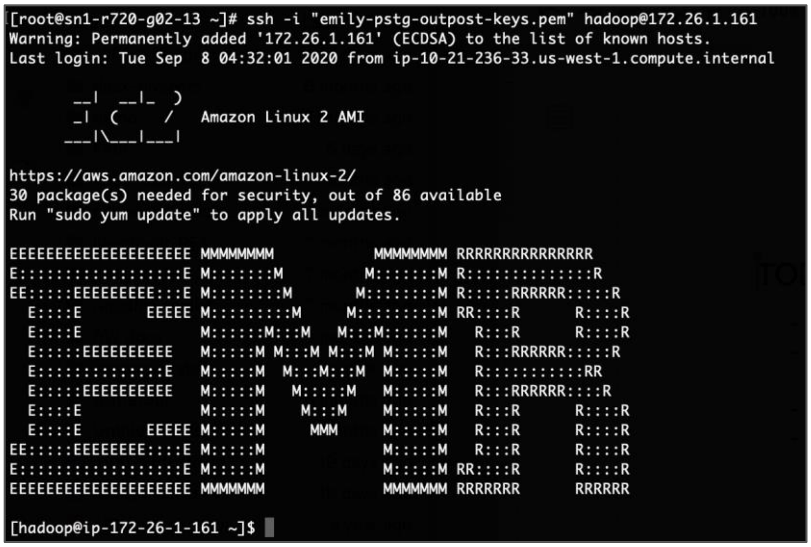
1. Open a terminal window. On Mac OS X, choose Applications > Utilities > Terminal. On other Linux distributions, terminal is typically found at Applications > Accessories > Terminal.
2. To establish a connection to the master node, type the following command. Replace ~/emily-pstg-outpost-keys.pem with the location and filename of the private key file (.pem) used to launch the cluster.

```
ssh -i ~/emily-pstg-outpost-keys.pem hadoop@ip-172-26-1-161.us-west-1.compute.internal
```

3. Type yes to dismiss the security warning.

Figure 17.

From a terminal, log into the master node via that command. Note that the login user for this cluster is named “hadoop” as opposed to the usual “ec2-user” or “ubuntu” users for standard EC2 instances.



```
[root@sn1-r720-g02-13 ~]# ssh -i "emily-pstg-outpost-keys.pem" hadoop@172.26.1.161
Warning: Permanently added '172.26.1.161' (ECDSA) to the list of known hosts.
Last login: Tue Sep  8 04:32:01 2020 from ip-10-21-236-33.us-west-1.compute.internal

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
30 package(s) needed for security, out of 86 available
Run "sudo yum update" to apply all updates.

EEEEEEEEEEEEEEEEEEEE MMMMMMMM           MMMMMMMM RRRRRRRRRRRRRRR
E::::::::::::::::::E M:::::::M         M:::::::M R::::::::::::::R
EE:::::EEEEEEEEE:::E M::::::::M       M::::::::M R:::::RRRRRR:::::R
  E::::E       EEEEE M:::::::::M     M:::::::::M RR::::R      R::::R
  E::::E             M::::::M:::M   M:::M::::::M   R:::R      R::::R
  E:::::EEEEEEEEEE   M:::::M M:::M M:::M M:::::M   R:::RRRRRR:::::R
  E::::::::::::::E   M:::::M  M:::M:::M  M:::::M   R:::::::::::RR
  E:::::EEEEEEEEEE   M:::::M   M:::::M   M:::::M   R:::RRRRRR::::R
  E::::E             M:::::M    M:::M    M:::::M   R:::R      R::::R
  E::::E       EEEEE M:::::M     MMM     M:::::M   R:::R      R::::R
EE:::::EEEEEEEE::::E M:::::M             M:::::M   R:::R      R::::R
E::::::::::::::::::E M:::::M             M:::::M RR::::R      R::::R
EEEEEEEEEEEEEEEEEEEE MMMMMMM             MMMMMMM RRRRRRR      RRRRRR

[hadoop@ip-172-26-1-161 ~]$
```

To access objects on FlashBlade S3 from this cluster, submit a job with the FlashBlade endpoint and credentials included. Example:

```
spark-submit --conf spark.hadoop.fs.s3a.endpoint=http://10.21.239.11 \
             --conf "spark.hadoop.fs.s3a.access.key=#########################################" \
             --conf "spark.hadoop.fs.s3a.secret.key=#########################################" \
             --master yarn \
             --deploy-mode cluster \
             wordcount.py \
             s3a://emily-outpost-bucket/sample.txt
```

Alternatively, document the FlashBlade specs in the spark-defaults.conf file so they're used by default automatically.

```
vi /etc/spark/conf.dist/spark-defaults.conf
```

Add the following lines to the bottom of the file:

```
spark.hadoop.fs.s3a.endpoint 10.21.239.11  # YOUR FLASHBLADE DATA VIP
spark.hadoop.fs.s3a.access.key=#########################################
spark.hadoop.fs.s3a.secret.key=#########################################

# Suggested tuning for FlashBlade performance.
spark.hadoop.fs.s3a.fast.upload true
spark.hadoop.fs.s3a.connection.ssl.enabled false
spark.hadoop.mapreduce.fileoutputcommitter.algorithm.version 2
spark.hadoop.mapreduce.input.fileinputformat.split.minsize 541073408
```

You can now submit jobs without specifying the FlashBlade specs in-line:

```
spark-submit --master yarn \
             --deploy-mode cluster \
             wordcount.py \
             s3a://emily-outpost-bucket/sample.txt
```

Performance of AWS Services backed by FlashBlade

Once again, we ran the S3-benchmark latency tests, and the results were in line with the previous section. As the pipeline becomes more complex and layers are added, latency mustn't be significantly impacted. FlashBlade can deliver low-latency to a range of AWS services running inside the Outpost.

Equally as important as the performance view from within each application is the view across applications. FlashBlade is performant across the range of IO profiles from the different applications running inside the Outpost, enabling developers to change out pipeline components without worrying about performance degradation caused by a change in the IO profile. FlashBlade is built from the ground up to consolidate all data-intensive applications by unifying file and object on a single scale-out platform, serving data to the broad range of applications in both on-prem and hybrid cloud locations.

Conclusions and Next Steps

The hybrid cloud infrastructure solution with FlashBlade and AWS Outposts brings cloud-adjacent scalable file and object performance to private and hybrid cloud deployments. The solution delivers all the benefits of FlashBlade, Purity, and the Pure1 architecture to AWS Outposts and users looking for end-to-end hybrid capabilities with low latency to meet the needs of modern data and applications. Pure Storage is focused on providing the cloud-like simplicity that fits well with AWS Outposts.

The reference architecture and testing demonstrate how new workloads, solutions, and scenarios are enabled when combining FlashBlade with AWS Outposts as a part of a hybrid data center strategy, with the performance that Pure's customers expect. The journey is just getting started. Our RA demonstrated a unique solution with simplicity, scale, and performance for diverse modern workloads such as AI and Analytics. Modern Data Protection, EDA, multi-cloud, and container management are areas where FlashBlade can bring value to AWS Outposts, and these topics will be explored in future reference architectures. Pure Storage is committed to providing premier, high-performance storage solutions for AWS Outposts now and in the future.

The Pure Storage Solutions Engineering Lab containing AWS Outposts is available to our user community and our partners for testing and performance validation of their solutions. If there is an interest in jointly conducting any testing, please contact aws@purestorage.com.

Additional Resources

- For more information about the Pure Storage partnership with AWS, visit Pure Storage Technology Partners: AWS.
- For more details on FlashBlade in an AI environment, visit Pure Storage AI Solutions.

THIS DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID. PURE STORAGE SHALL NOT BE LIABLE FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS DOCUMENTATION. THE INFORMATION CONTAINED IN THIS DOCUMENTATION IS SUBJECT TO CHANGE WITHOUT NOTICE.

Pure Storage, Inc.
650 Castro Street, #400
Mountain View, CA 94041