AWS New User Orientation

Robert Wang, Ph.D.
Solutions Architect, Higher Education
wangrob@amazon.com

Pillars of AWS Well-Architected

Well-Architected Labs: https://wellarchitectedlabs.com/

Operational Excellence

Security

Reliability

Performance Efficiency

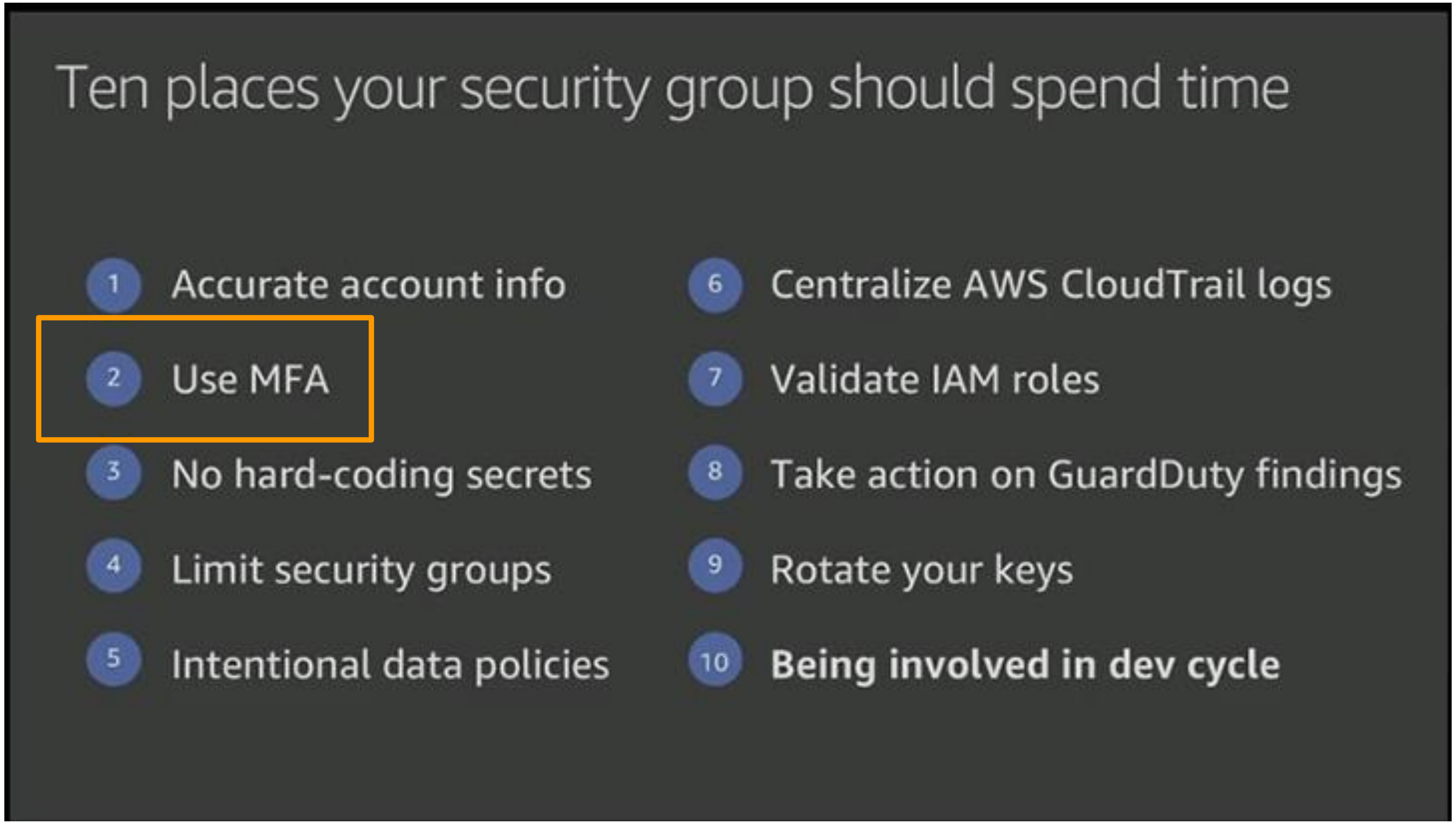
Cost Optimization

Sustainability

Account Security

AWS Account Security Best Practices

Reference: https://aws.amazon.com/blogs/security/top-10-security-items-to-improve-in-your-aws-account/



Ten places your security group should spend time

1 Accurate account info

2 Use MFA

3 No hard-coding secrets

4 Limit security groups

5 Intentional data policies

6 Centralize AWS CloudTrail logs

7 Validate IAM roles

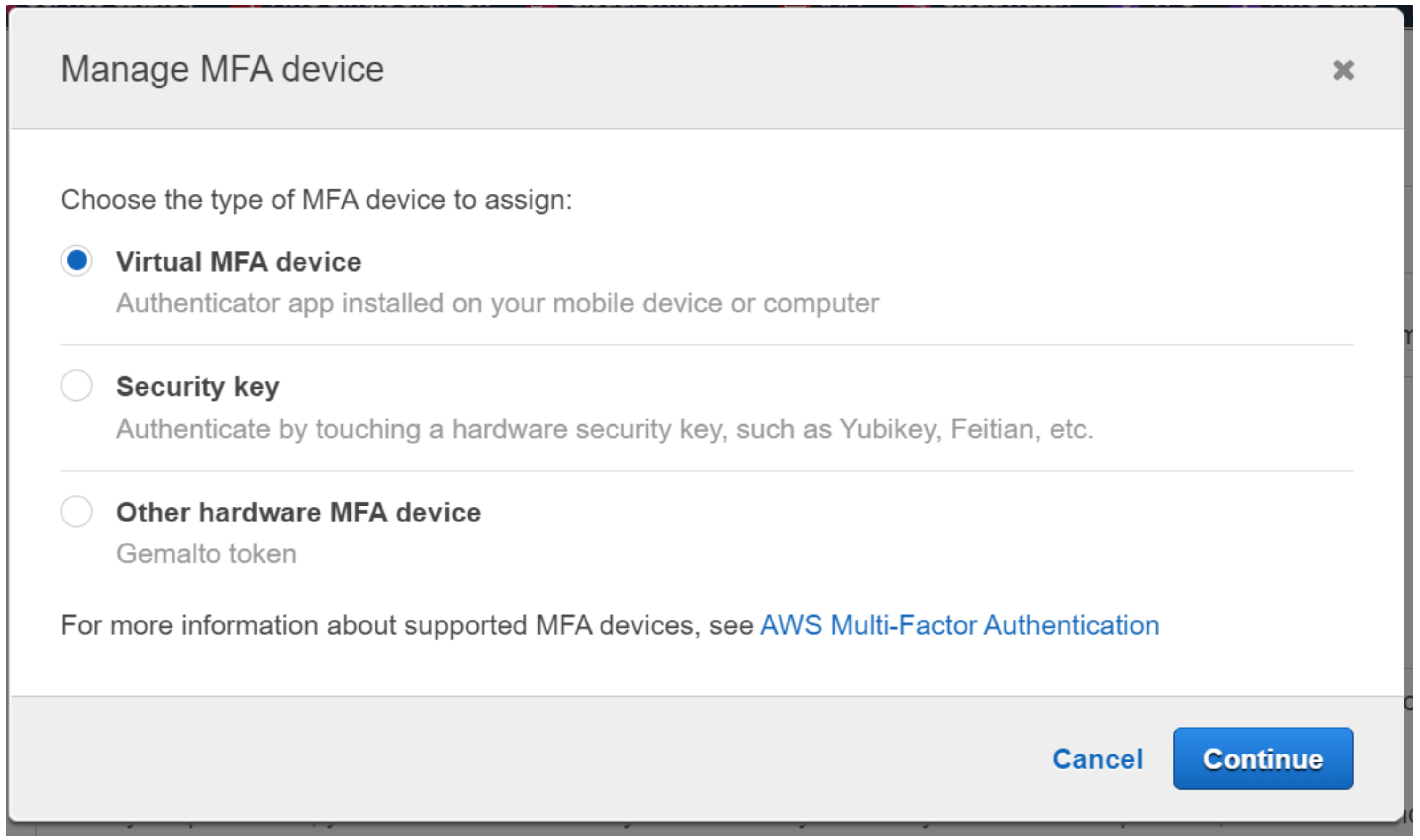
8 Take action on GuardDuty findings

9 Rotate your keys

10 **Being involved in dev cycle**

Enable MFA for Root User and IAM User

IAM > IAM User > Security Credentials > Assigned MFA device > Manage:



Manage MFA device

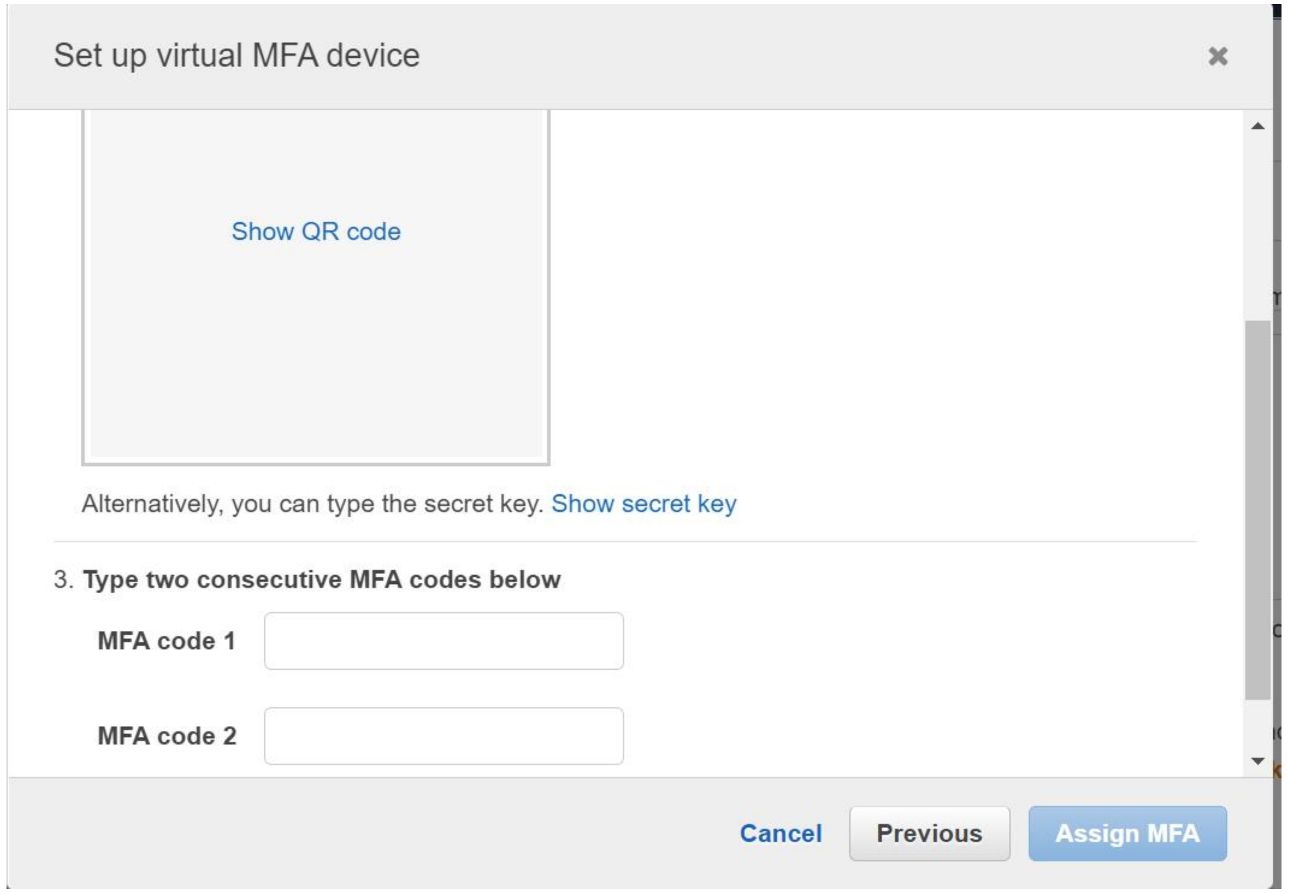
Choose the type of MFA device to assign:

[✓ **Virtual MFA device**]
Authenticator app installed on your mobile device or computer

Security key
Authenticate by touching a hardware security key, such as Yubikey, Feitian, etc.

Other hardware MFA device
Gemalto token

For more information about supported MFA devices, see AWS Multi-Factor Authentication



Set up virtual MFA device

Alternatively, you can type the secret key. Show secret key

3. **Type two consecutive MFA codes below**

MFA code 1	
MFA code 2	

Amazon S3 Security Best Practices

Top 10: https://aws.amazon.com/blogs/security/top-10-security-best-practices-for-securing-data-in-amazon-s3/

Docs:
https://docs.aws.amazon.com/AmazonS3/latest/userguide/security-best-practices.html

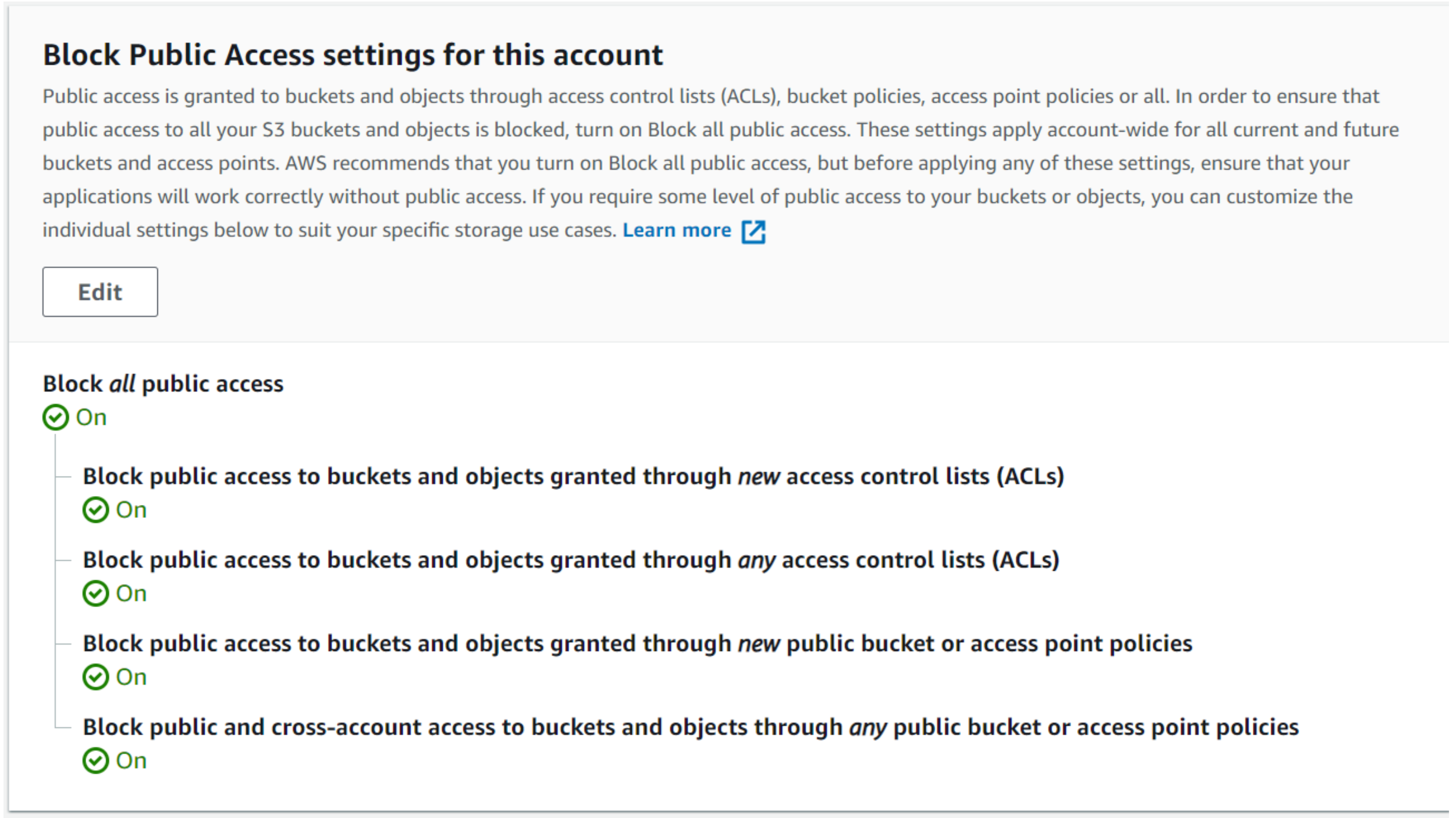
Encryption at-rest: Server-side encryption:
https://docs.aws.amazon.com/AmazonS3/latest/userguide/serv-side-encryption.html

- **SSE-S3 vs. KMS: When is KMS required?**

Enforce KMS encryption at-rest during object upload:
https://aws.amazon.com/premiumsupport/knowledge-center/s3-bucket-store-kms-encrypted-objects/

Enforce encryption in-transit during object upload:
https://aws.amazon.com/premiumsupport/knowledge-center/s3-bucket-store-kms-encrypted-objects/

https://s3.console.aws.amazon.com/s3/settings



Block Public Access settings for this account

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies or all. In order to ensure that public access to all your S3 buckets and objects is blocked, turn on Block all public access. These settings apply account-wide for all current and future buckets and access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to your buckets or objects, you can customize the individual settings below to suit your specific storage use cases. Learn more

Block *all* public access
✓ On

- Block public access to buckets and objects granted through *new* access control lists (ACLs)
  ✓ On
- Block public access to buckets and objects granted through *any* access control lists (ACLs)
  ✓ On
- Block public access to buckets and objects granted through *new* public bucket or access point policies
  ✓ On
- Block public and cross-account access to buckets and objects through *any* public bucket or access point policies
  ✓ On

Create IAM User Groups and Users

Create IAM User Group

- IAM > User Groups
- Create group, e.g.:
- "Admin" group
    - Name: Admin
    - Permission: AdministratorAccess
- "Developers" group
    - Name: Developers
    - Permission: AmazonS3Access

Create IAM User

- IAM > Users
- Create user, e.g.:
- Add "robert" to "Admin" group
    - Name: robert
    - Prog. access: Check
    - Console access: Check
    - Set password; disable require new password
    - Add to "Admin" group
- **NOTE: Be sure to download and save user credentials csv file!**

Install and configure AWS Command Line Tool (CLI) v2

Install AWS CLI v2

- Installation instructions: https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html
- Linux, Windows (Git Bash)
    - Open terminal
    - $ cd
    - $ mkdir installers
    - Execute installation steps
    - Verify:
    - $ aws --version

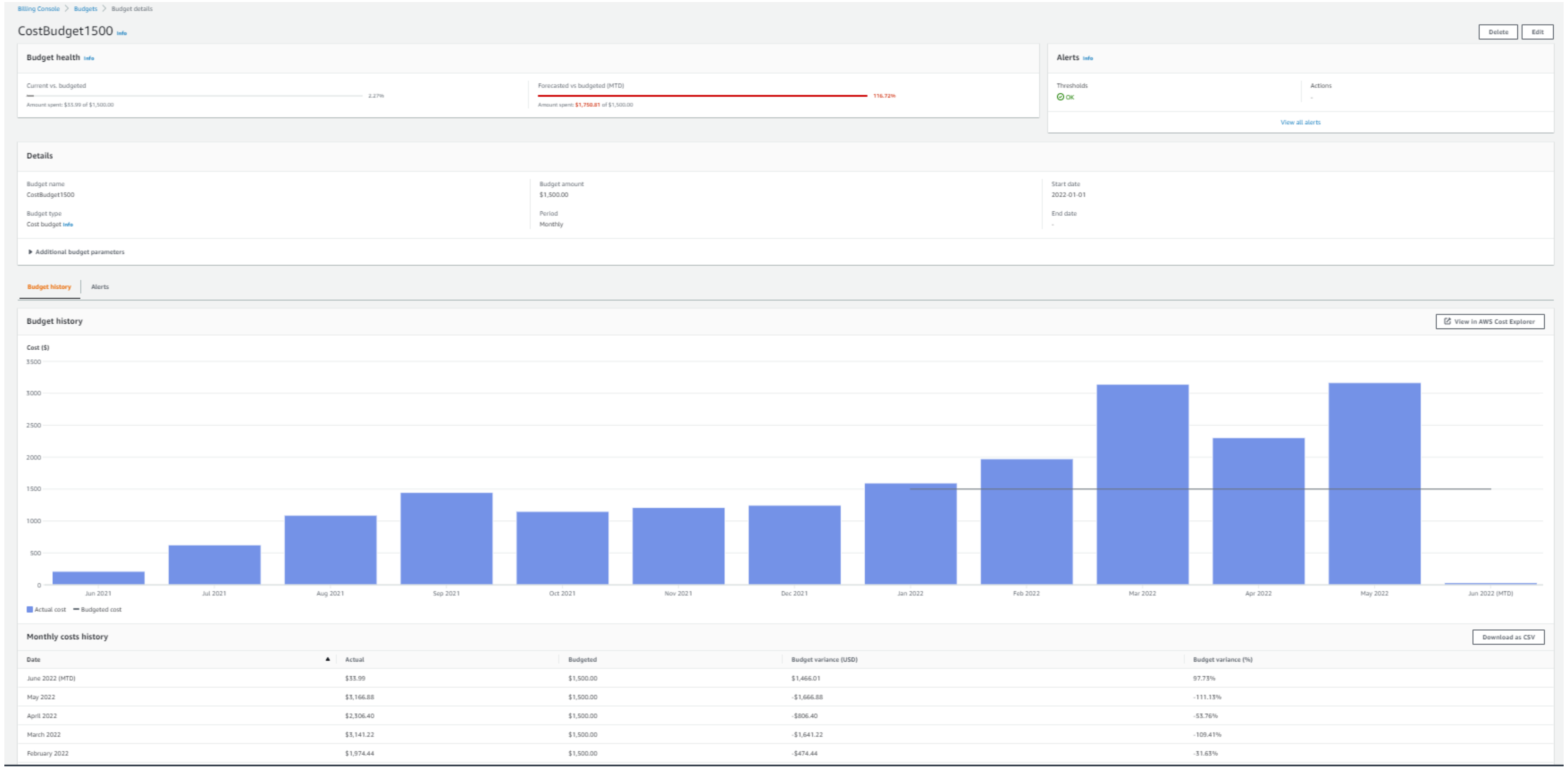
Configure AWS CLI v2

- $ aws configure (--profile default)
- AWS Access Key Id: (Enter value from user credentials csv file.)
- AWS Secret Access Key: (Enter value from user credentials csv.)
- Default region name: e.g., us-west-2, us-east-1
- Default output format: json
- Verify:
- $ aws s3 ls

Cost Control

Budgets (Budget Alerts)

https://console.aws.amazon.com/billing/home#/budgets/overview



CostBudget1500

Budget health

Current vs. budgeted — 2.27%
Amount spent: $33.99 of $1,500.00

Forecasted vs budgeted (MTD) — 116.72%
Amount spent: $1,750.81 of $1,500.00

Alerts

Thresholds	Actions
✓ OK	-

Details

Budget name: CostBudget1500	Budget amount: $1,500.00	Start date: 2022-01-01
Budget type: Cost budget	Period: Monthly	End date: -

▸ Additional budget parameters

Budget history

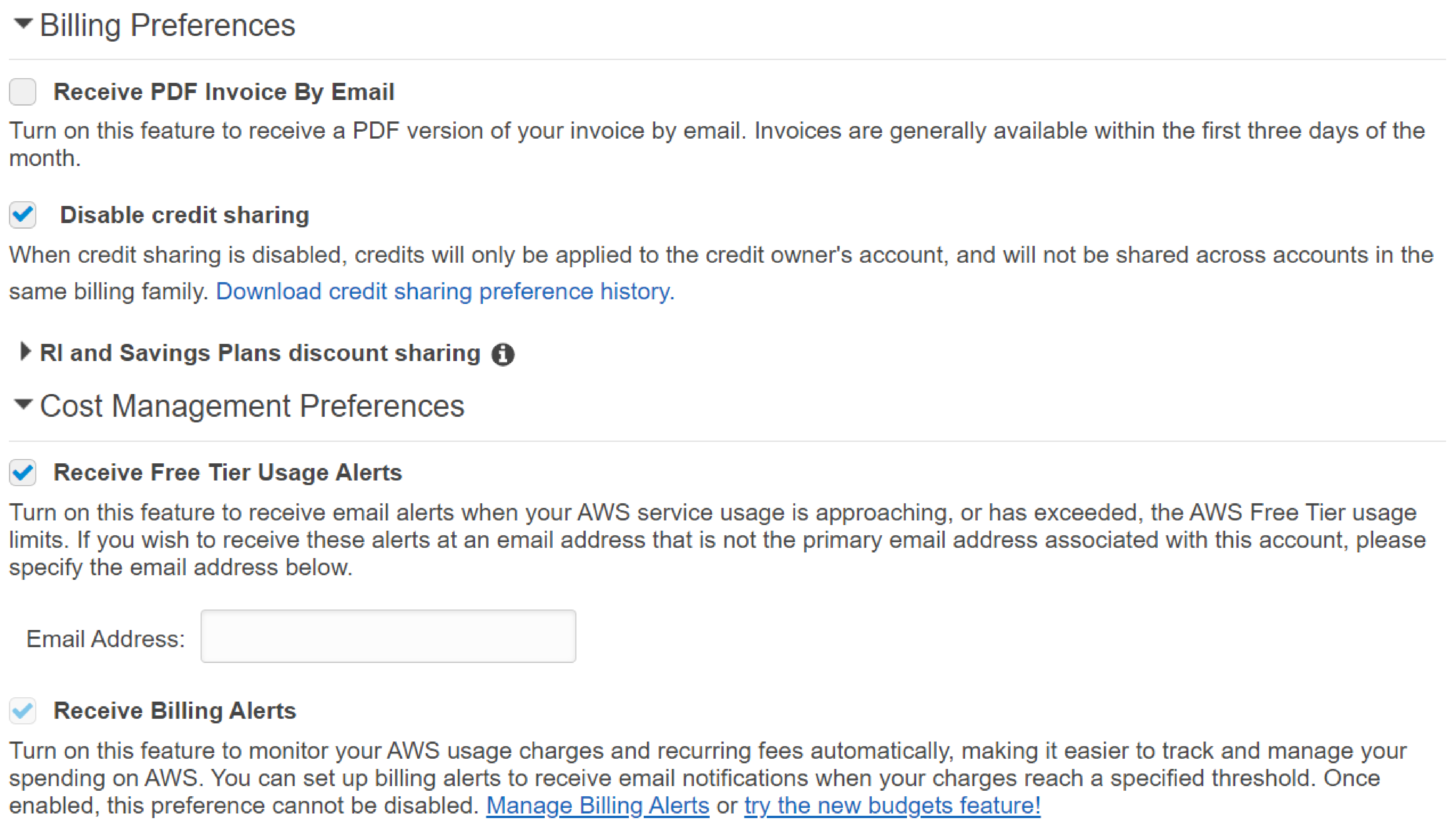
■ Actual cost — Budgeted cost

Monthly costs history

Date ▲	Actual	Budgeted	Budget variance (USD)	Budget variance (%)
June 2022 (MTD)	$33.99	$1,500.00	$1,466.01	97.73%
May 2022	$3,166.88	$1,500.00	-$1,666.88	-111.13%
April 2022	$2,306.40	$1,500.00	-$806.40	-53.76%
March 2022	$3,141.22	$1,500.00	-$1,641.22	-109.41%
February 2022	$1,974.44	$1,500.00	-$474.44	-31.63%

Billing Preferences

https://console.aws.amazon.com/billing/home?#/preferences



▼ Billing Preferences

☐ **Receive PDF Invoice By Email**

Turn on this feature to receive a PDF version of your invoice by email. Invoices are generally available within the first three days of the month.

☑ **Disable credit sharing**

When credit sharing is disabled, credits will only be applied to the credit owner's account, and will not be shared across accounts in the same billing family. Download credit sharing preference history.

▶ **RI and Savings Plans discount sharing**

▼ Cost Management Preferences

☑ **Receive Free Tier Usage Alerts**

Turn on this feature to receive email alerts when your AWS service usage is approaching, or has exceeded, the AWS Free Tier usage limits. If you wish to receive these alerts at an email address that is not the primary email address associated with this account, please specify the email address below.

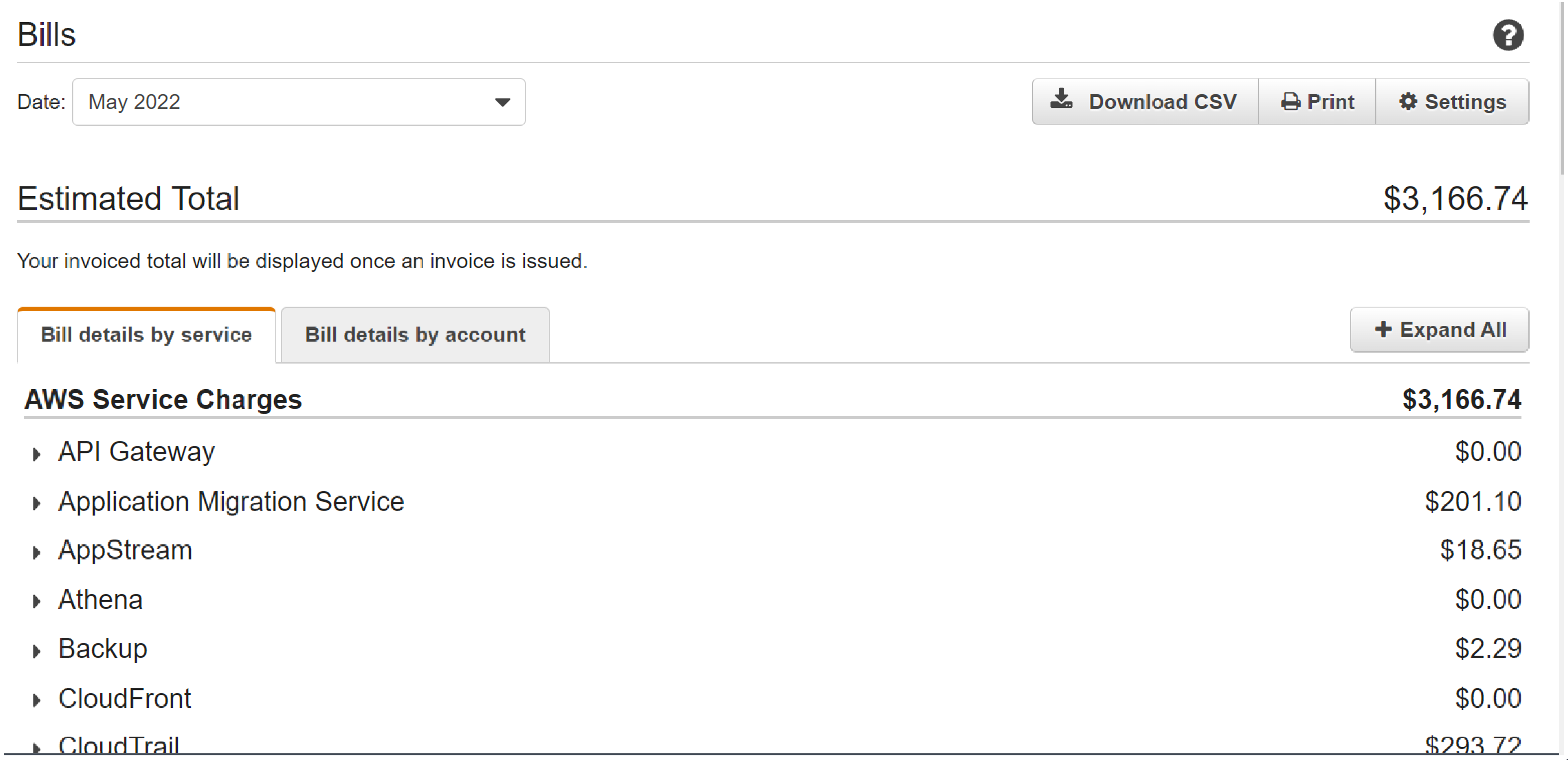
Email Address:

☑ **Receive Billing Alerts**

Turn on this feature to monitor your AWS usage charges and recurring fees automatically, making it easier to track and manage your spending on AWS. You can set up billing alerts to receive email notifications when your charges reach a specified threshold. Once enabled, this preference cannot be disabled. Manage Billing Alerts or try the new budgets feature!

Billing Dashboard

https://console.aws.amazon.com/billing/home?#/bills



Bills

Date: May 2022

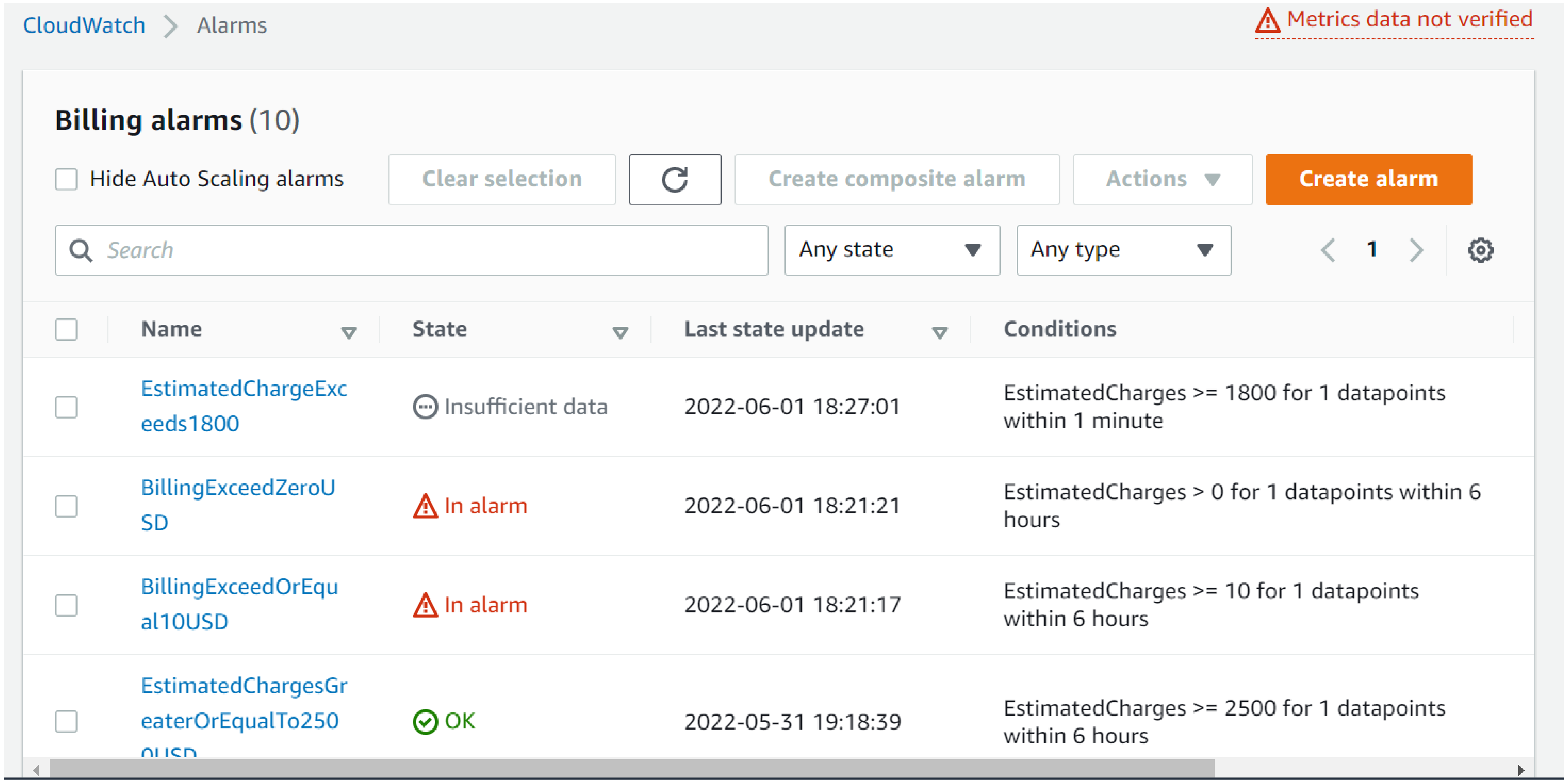
Estimated Total	$3,166.74

Your invoiced total will be displayed once an invoice is issued.

AWS Service Charges	$3,166.74
API Gateway	$0.00
Application Migration Service	$201.10
AppStream	$18.65
Athena	$0.00
Backup	$2.29
CloudFront	$0.00
CloudTrail	$293.72

Billing Alarms

https://console.aws.amazon.com/cloudwatch/home?#alarmsV2:?~(namespace~'AWS*2fBilling)


CloudWatch > Alarms

Metrics data not verified

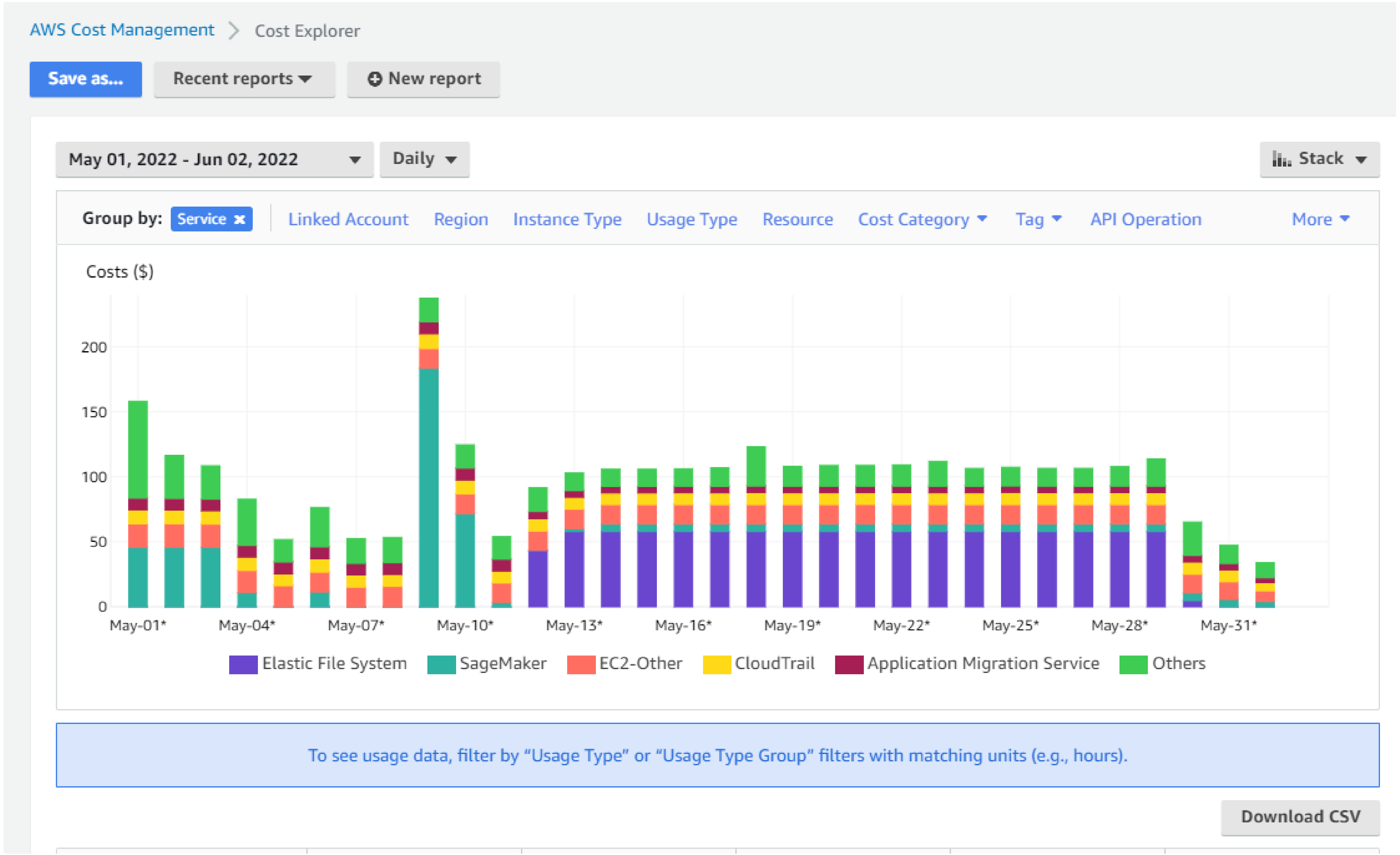
Billing alarms (10)

Hide Auto Scaling alarms

	Name	State	Last state update	Conditions
☐	EstimatedChargeExceeds1800	Insufficient data	2022-06-01 18:27:01	EstimatedCharges >= 1800 for 1 datapoints within 1 minute
☐	BillingExceedZeroUSD	In alarm	2022-06-01 18:21:21	EstimatedCharges > 0 for 1 datapoints within 6 hours
☐	BillingExceedOrEqual10USD	In alarm	2022-06-01 18:21:17	EstimatedCharges >= 10 for 1 datapoints within 6 hours
☐	EstimatedChargesGreaterOrEqualTo2500USD	OK	2022-05-31 19:18:39	EstimatedCharges >= 2500 for 1 datapoints within 6 hours

Cost Management (Cost Explorer)

https://console.aws.amazon.com/cost-management/home#/dashboard



May 01, 2022 - Jun 02, 2022 · Daily

Group by: Service · Linked Account · Region · Instance Type · Usage Type · Resource · Cost Category · Tag · API Operation · More

Costs ($)

To see usage data, filter by "Usage Type" or "Usage Type Group" filters with matching units (e.g., hours).

AWS Credits

Eligible Services

https://aws.amazon.com/awscredits/

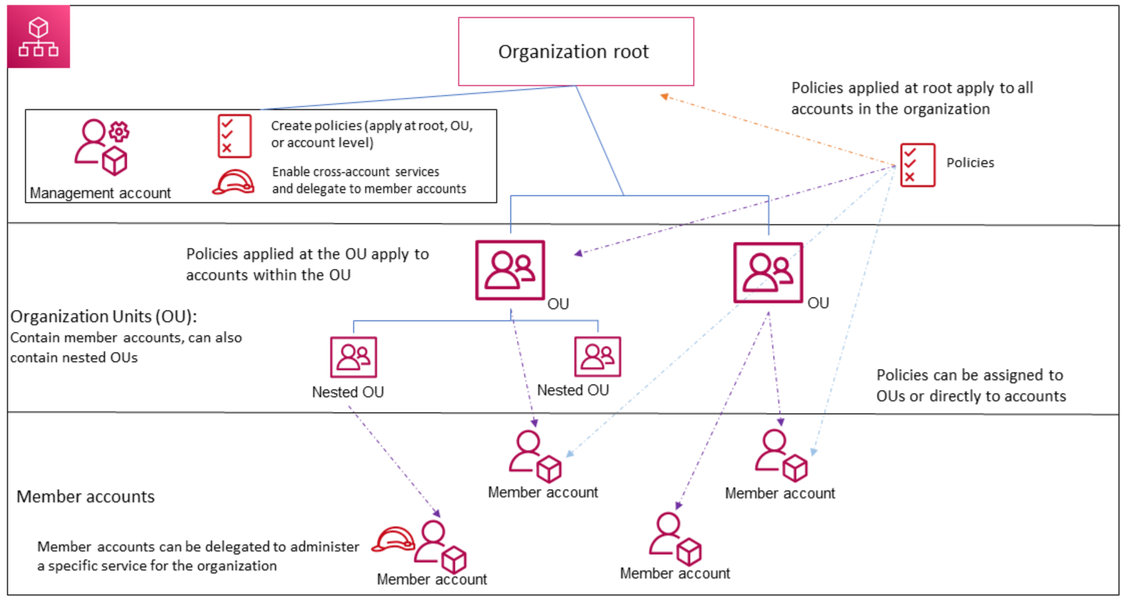
1. Promotional Credit will be applied only to offset eligible fees and charges incurred during or following the billing cycle in which you apply the applicable Promotional Credit code to your AWS account. Promotional Credit will be applied only to the specific Services designated by your AWS contracting entity (collectively, “Eligible Services”). **Promotional Credit will not be applied to any fees or charges for Amazon Mechanical Turk, AWS Managed Services, Ineligible AWS Support, AWS Marketplace, AWS Professional Services, AWS Training, AWS Certification, Amazon Route 53 domain name registration or transfer, any Services for mining for cryptocurrency, any other Services as may be designated by your AWS contracting entity, or any upfront fee for any Services such as Savings Plans and Reserved Instances (collectively, “Ineligible Services”).** For purposes of this Section 1, “Ineligible AWS Support” means AWS Support (as described on the AWS Site) that is at the Enterprise Support level.

Sharing AWS Credits: AWS Organization (1)

- **Creating an Organization:** https://docs.aws.amazon.com/organizations/latest/userguide/orgs_tutorials_basic.html#tut-basic-prereqs

- **Inviting an Account:** https://docs.aws.amazon.com/organizations/latest/userguide/orgs_manage_accounts_invites.html

- **Removing an Account:** https://docs.aws.amazon.com/organizations/latest/userguide/orgs_manage_accounts_remove.html


Organization root
Policies applied at root apply to all accounts in the organization
Management account
Create policies (apply at root, OU, or account level)
Enable cross-account services and delegate to member accounts
Policies
Policies applied at the OU apply to accounts within the OU
OU
OU
Organization Units (OU):
Contain member accounts, can also contain nested OUs
Nested OU
Nested OU
Policies can be assigned to OUs or directly to accounts
Member accounts
Member account
Member account
Member accounts can be delegated to administer a specific service for the organization
Member account
Member account

- **Sharing AWS Credits:** https://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/useconsolidatedbilling-credits.html#credit-sharing

- **How are credits calculated:** https://aws.amazon.com/premiumsupport/knowledge-center/consolidated-billing-credits/

Sharing AWS Credits: AWS Organization (2)

Strategies

1. Create AWS Organizations now
2. Create AWS Organizations later

Set up

- Select or create a new Payer Account
- Create AWS Organizations
- Invite all member accounts
- Enable credit sharing

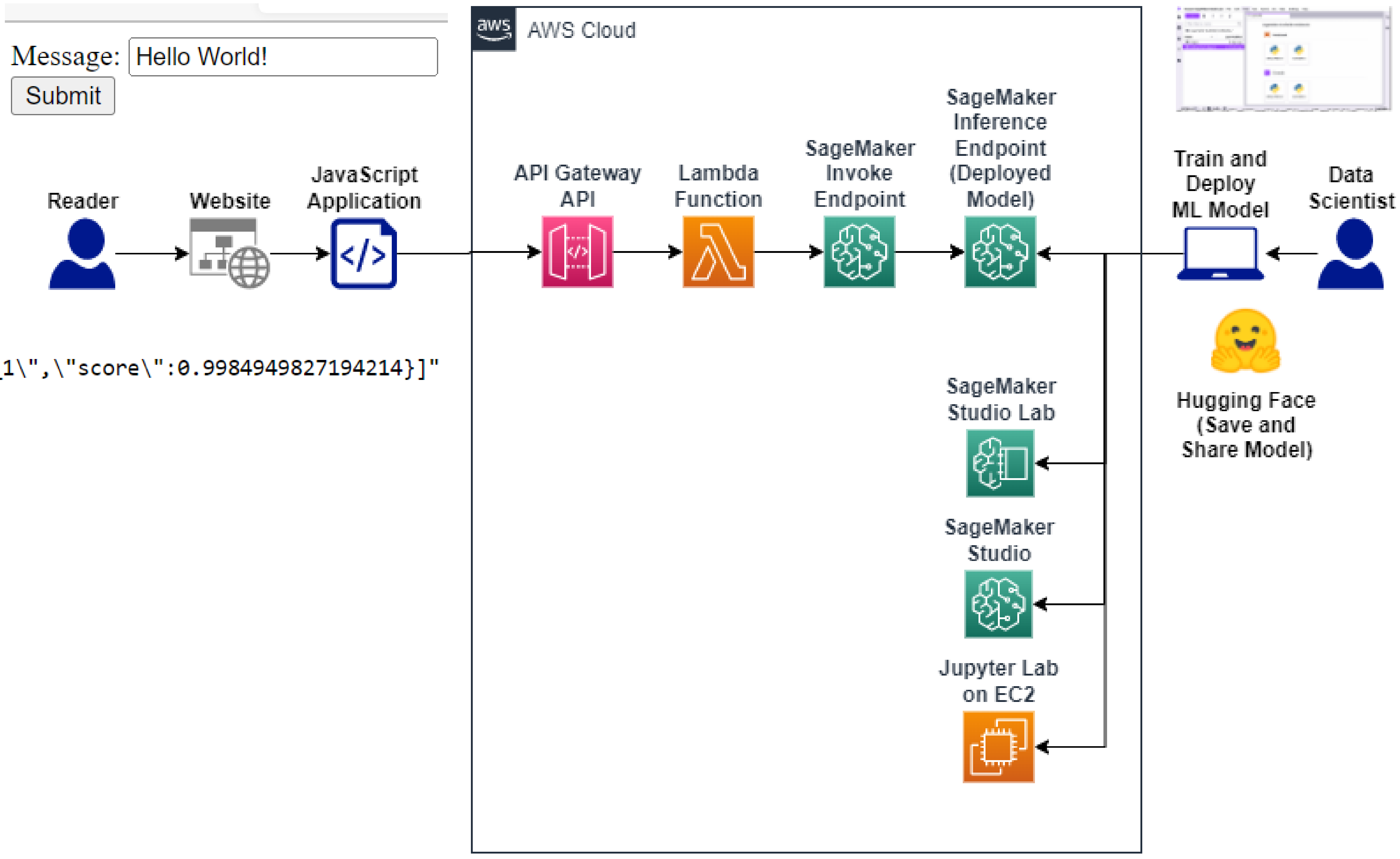
How are credits calculated:
https://aws.amazon.com/premiumsupport/knowledge-center/consolidated-billing-credits/

Clean up

- Remove all member accounts
- Credits will remain with the Payer Account!

Text Classification Project Architecture

Text Classification Project Architecture


Message: Hello World!

Submit

```
"[{\"label\":\"LABEL_1\",\"score\":0.9984949827194214}]"
```

Training Notebooks that save model to Hugging Face

Text Classification: sst2

- PyTorch: sst2_text_classification.ipynb
  - SMS Lab: 6 hours
  - SMS or EC2: <1 hour
- TensorFlow: sst2_text_classification-tf.ipynb
  - SMS Lab: 4.5 hours
  - SMS or EC2: <1 hour

Image Classification: vit

- PyTorch: vit_image_classification.ipynb
  - SMS Lab: Exceed time limit
  - SMS or EC2: <1 hour
- TensorFlow: vit_image_classification-tf.ipynb
  - SMS Lab: Exceed time limit
  - SMS or EC2: 1.5 hour

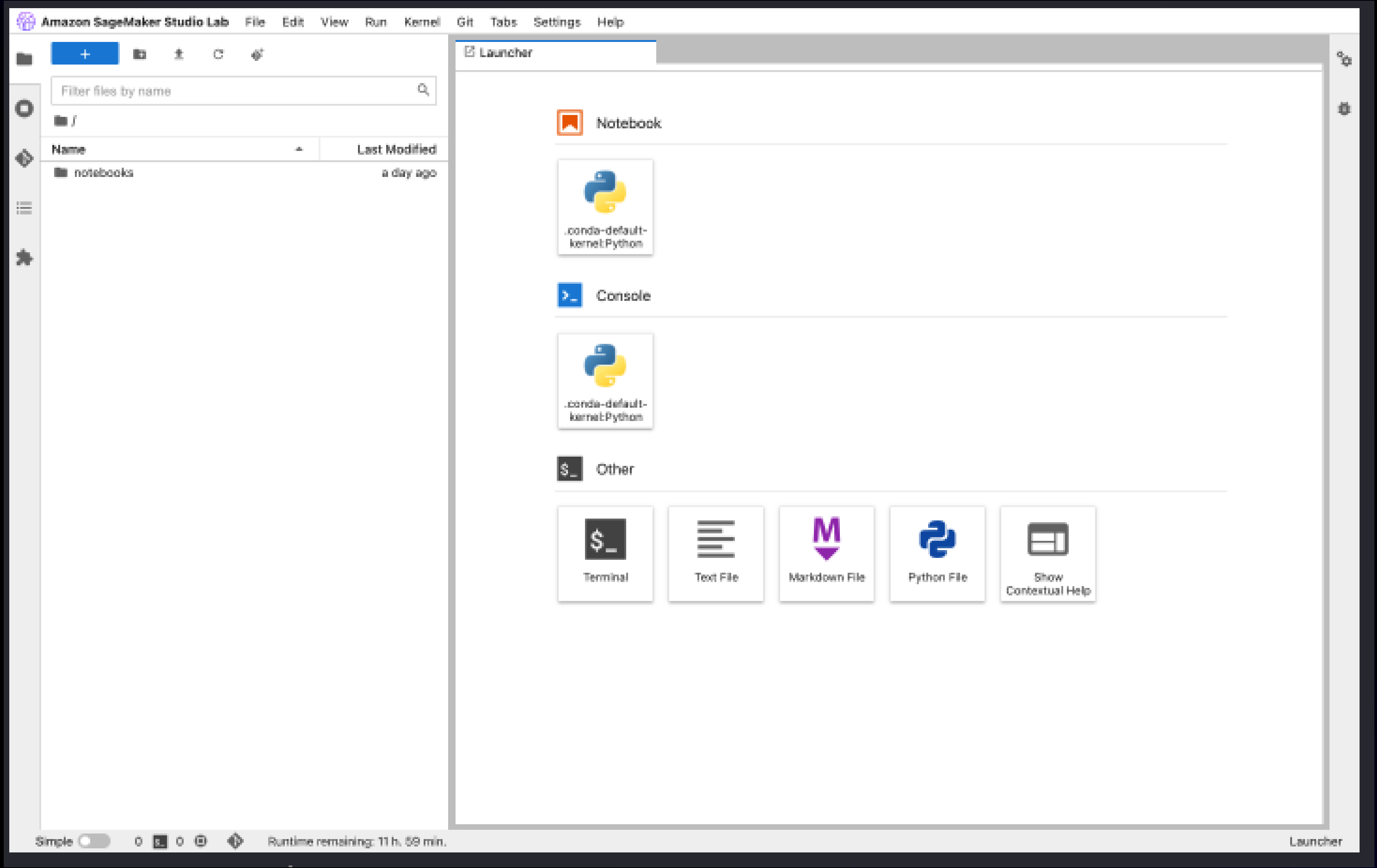
Even for image classification, as a pre-integrated Jupypter Lab environment, SageMaker Studio Lab is a great tool. SMSL requires no setup and it's FREE!

SageMaker Studio Lab and SageMaker Studio as Jupyter Lab Server

(Music Video: “Top Gun” Anthem with jets taking off from the deck of an aircraft carrier.)

Amazon SageMaker Studio Lab (FREE Service)

https://studiolab.sagemaker.aws/ : Referral Code: Email wangrob@amazon.com



Amazon SageMaker Studio Lab File Edit View Run Kernel Git Tabs Settings Help
Filter files by name
Name
Last Modified
notebooks
a day ago
Launcher
Notebook
.conda-default-kernel:Python
Console
.conda-default-kernel:Python
Other
Terminal
Text File
Markdown File
Python File
Show Contextual Help
Simple
Runtime remaining: 11 h. 59 min.
Launcher

- Familiar JupyterLab experience
  - Terminal access
  - Git/GitHub
- Your ML environment on AWS
  - Compute dedicated to you
  - 12 hours CPU/4 hours GPU
  - Install the libraries you want
  - Dedicated 15 GB for your project
  - Unlimited user sessions
  - Pick up where you left off

Launch SageMaker Studio Lab

https://studiolab.sagemaker.aws/ (Separate from your AWS Account)

Launch SageMaker Studio Lab

- Sign up for free account: https://studiolab.sagemaker.aws/
- Please use email address domain that matches referral code.
- Select CPU (or GPU).
- Click **Start runtime**.
- Click **Open project**.

Install and configure AWS CLI v2

- Please execute the steps in the next page.

Create Hugging Face account and Install git-lfs

- These are required for saving models to Hugging Face.

Install and configure AWS Command Line Tool (CLI) v2

Install AWS CLI v2

- Installation instructions: https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html
- Linux, Windows (Git Bash)
    - Open terminal
    - $ cd
    - $ mkdir installers
    - Execute installation steps
    - Verify:
    - $ aws --version

Configure AWS CLI v2

- $ aws configure (--profile default)
- AWS Access Key Id: (Enter value from user credentials csv file.)
- AWS Secret Access Key: (Enter value from user credentials csv.)
- Default region name: e.g., us-west-2, us-east-1
- Default output format: json
- Verify:
- $ aws s3 ls

Create Hugging Face account and install git-lfs (SM Studio Lab)

Create Hugging Face account

- Hugging Face: https://huggingface.co/
- Hugging Face is like a GitHub for ML models and datasets.
- Hugging Face has its own HuggingFace_Hub API for saving ("pushing") and retrieving models.
- Please create a Hugging Face account saving and retrieving models.
- **HF Transformers Notebooks: https://huggingface.co/docs/transformers/v4.16.2/en/notebooks**

Install Git Large File Storage (git-lfs)

- $ pip install git-lfs
- Verify: $ git-lfs --version

Download and Execute Test Notebooks

Download Test Notebooks

1) $ mkdir ~/workspace

2) $ cd ~/workspace

3) $ git clone https://github.com/rwang5688/sm-stack-v2.git

Execute List S3 Buckets Notebook

1) From file explorer, navigate to: ~/workspace/sm-stack-v2/sm-model/s3

2) Launch and run list_s3_buckets.ipynb

NOTE: This step assumes that you have used “aws configure” to set “default” profile.

Training Notebooks that save model to Hugging Face

Text Classification: sst2

- PyTorch: sst2_text_classification.ipynb
  - SMS Lab: 6 hours
  - SMS or EC2: <1 hour
- TensorFlow: sst2_text_classification-tf.ipynb
  - SMS Lab: 4.5 hours
  - SMS or EC2: <1 hour

Image Classification: vit

- PyTorch: vit_image_classification.ipynb
  - SMS Lab: Exceed time limit
  - SMS or EC2: <1 hour
- TensorFlow: vit_image_classification-tf.ipynb
  - SMS Lab: Exceed time limit
  - SMS or EC2: 1.5 hour

Even for image classification, as a pre-integrated Jupypter Lab environment, SageMaker Studio Lab is a great tool. SMSL requires no setup and it's FREE!

Amazon SageMaker Studio Best Practices (PAID Service)

- In order to use SageMaker Studio, you must set up a SageMaker Domain.
- Once you set up a SageMaker Domain, you are always paying for the domain's EFS file system storage.
- **Know your SageMaker service quotas: https://console.aws.amazon.com/servicequotas/home/services/sagemaker/quotas ...**
- **SageMaker service quota unit = Instances, NOT vCPUs!**
- **For SageMaker Studio as Jupyter Lab Server: Request "Studio KernelGateway Apps running on ml.${EC2.instance.type}", e.g., ml.g4dn.16xlarge (1 GPU, 64 vCPUs, 256 GiB).**
- Please make sure your SageMaker Studio applications and S3 buckets are in the same region (Note: Be ware of cross-region data transfer charge).
- SageMaker Studio applications are ALWAYS RUNNING.
- **THEREFORE** if you don't need a SageMaker Studio application, please DELETE it.

Launch SageMaker Studio (Set up SageMaker Domain)

https://console.aws.amazon.com/sagemaker/home (Part of your AWS Account)

Launch SageMaker Studio

- Please execute the steps in this tutorial: https://catalog.us-east-1.prod.workshops.aws/workshops/63069e26-921c-4ce1-9cc7-dd882ff62575/en-US/prerequisites/option2
- Steps include setting up SageMaker Domain.

Install and configure AWS CLI v2

- Please execute the steps in the next page.

Create Hugging Face account and Install git-lfs

- These are required for saving models to Hugging Face.

Install and configure AWS Command Line Tool (CLI) v2

Install AWS CLI v2

- Installation instructions: https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html
- Linux, Windows (Git Bash)
    - Open terminal
    - $ cd
    - $ mkdir installers
    - Execute installation steps
    - Verify:
    - $ aws --version

Configure AWS CLI v2

- $ aws configure (--profile default)
- AWS Access Key Id: (Enter value from user credentials csv file.)
- AWS Secret Access Key: (Enter value from user credentials csv.)
- Default region name: e.g., us-west-2, us-east-1
- Default output format: json
- Verify:
- $ aws s3 ls

Create Hugging Face account and install git-lfs (SM Studio)

Create Hugging Face account

- Hugging Face: https://huggingface.co/
- Hugging Face is like a GitHub for ML models and datasets.
- Hugging Face has its own HuggingFace_Hub API for saving ("pushing") and retrieving models.
- Please create a Hugging Face account saving and retrieving models.
- **HF Transformers Notebooks: https://huggingface.co/docs/transformers/v4.16.2/en/notebooks**

Install Git Large File Storage (git-lfs)

- Navigate to git-lfs: https://git-lfs.github.com/
- Click **Downloads**
- Scroll down to "Assets"
- Copy link for "Linux AMD64"
- Open "System Terminal"
- (Same instructions for EC2 Linux)
- $ cd installers
- $ wget ${link for Linux AMD64}
- $ bash git-lfs-${version number}/install.sh

NOTE: Need to repeat the last step for every "Image Terminal".

Download and Execute Test Notebooks

Download Test Notebooks

1) $ mkdir ~/workspace

2) $ cd ~/workspace

3) $ git clone https://github.com/rwang5688/sm-stack-v2.git

Execute List S3 Buckets Notebook

1) From file explorer, navigate to: ~/workspace/sm-stack-v2/sm-model/s3

2) Launch and run list_s3_buckets.ipynb

NOTE: This step assumes that you have used “aws configure” to set “default” profile.

Training Notebooks that save model to Hugging Face

Text Classification: sst2

- PyTorch: sst2_text_classification.ipynb
    - SMS Lab: 6 hours
    - SMS or EC2: <1 hour
- TensorFlow: sst2_text_classification-tf.ipynb
    - SMS Lab: 4.5 hours
    - SMS or EC2: <1 hour

Image Classification: vit

- PyTorch: vit_image_classification.ipynb
    - SMS Lab: Exceed time limit
    - SMS or EC2: <1 hour
- TensorFlow: vit_image_classification-tf.ipynb
    - SMS Lab: Exceed time limit
    - SMS or EC2: 1.5 hour

Even for image classification, as a pre-integrated Jupypter Lab environment, SageMaker Studio Lab is a great tool. SMSL requires no setup and it's FREE!

Set up Jupyter Lab Server on EC2 Instance

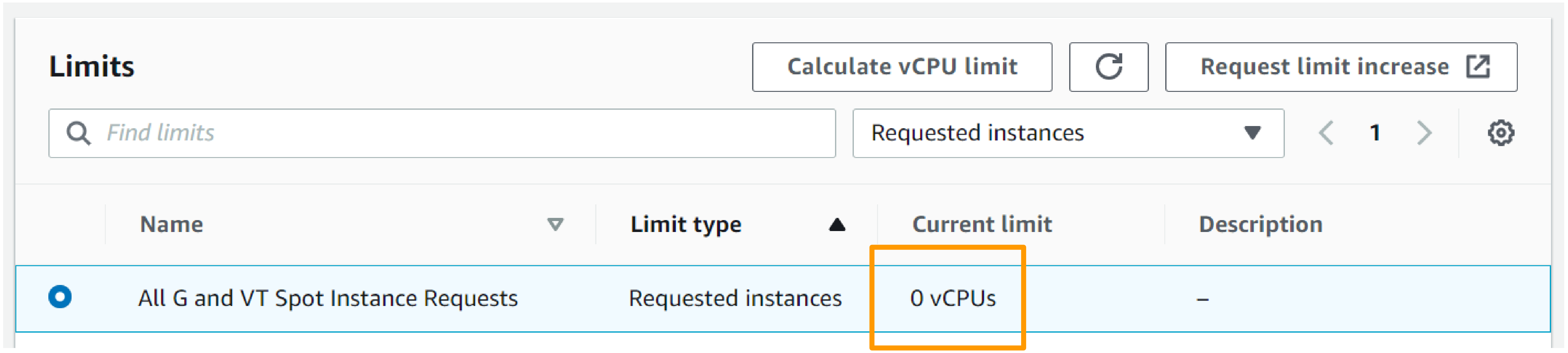
(Music Video: “Hold My Hand” with propeller plane taking off from your garage drive way.)

Amazon EC2 Best Practices (PAID Service)

- **Know your EC2 service limits:** **https://console.aws.amazon.com/ec2/v2/home?#Limits …**
- **EC2 service limit unit = vCPUs, NOT Instances!**
- **A.k.a. EC2 service quota:** **https://console.aws.amazon.com/servicequotas/home/services/ec2/quotas …**
- Please make sure your EC2 instances and S3 buckets are in the same region (Note: Be ware of cross-region data transfer charge).
- You don’t pay for stopped EC2 instances.
- BUT You are always paying for EBS volume (disk) storage and Elastic IP (static IP) that you allocate.
- **THEREFORE** if you don’t need an EC2 instance, please TERMINATE it.
- If you want to keep your customization, you can create Custom Amazon Machine Images (AMIs): https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/creating-an-ami-ebs.html …

EC2 Service Limit Increase: How to request

Step 1: You must request limit increase and be DENIED or PARTIALLY APPROVED.
https://console.aws.amazon.com/ec2/v2/home?#Limits … Unit = vCPUs, NOT Instances!



Name	Limit type	Current limit	Description
All G and VT Spot Instance Requests	Requested instances	0 vCPUs	–

Step 2: If DENIED or PARTIALLY APPROVED, SA can escalate on your behalf.

- The following information/template is **REQUIRED** in your ticket. If any of this is missing, the request will be denied and ticket resolved with no action.
  - **Account ID -**
  - **Support Case ID/link -**
  - **Previous denied/partially approved SQIR link -**
  - **NEW Pending SLI request for us to review -**
  - **Sales/BDs/TAMs/SAs/Executive asking for escalation review -**
  - **Promotional credits (if applicable) -**
  - **Any other unique considerations or use case we should take in to account -**

Create and connect to EC2 Instance (1)

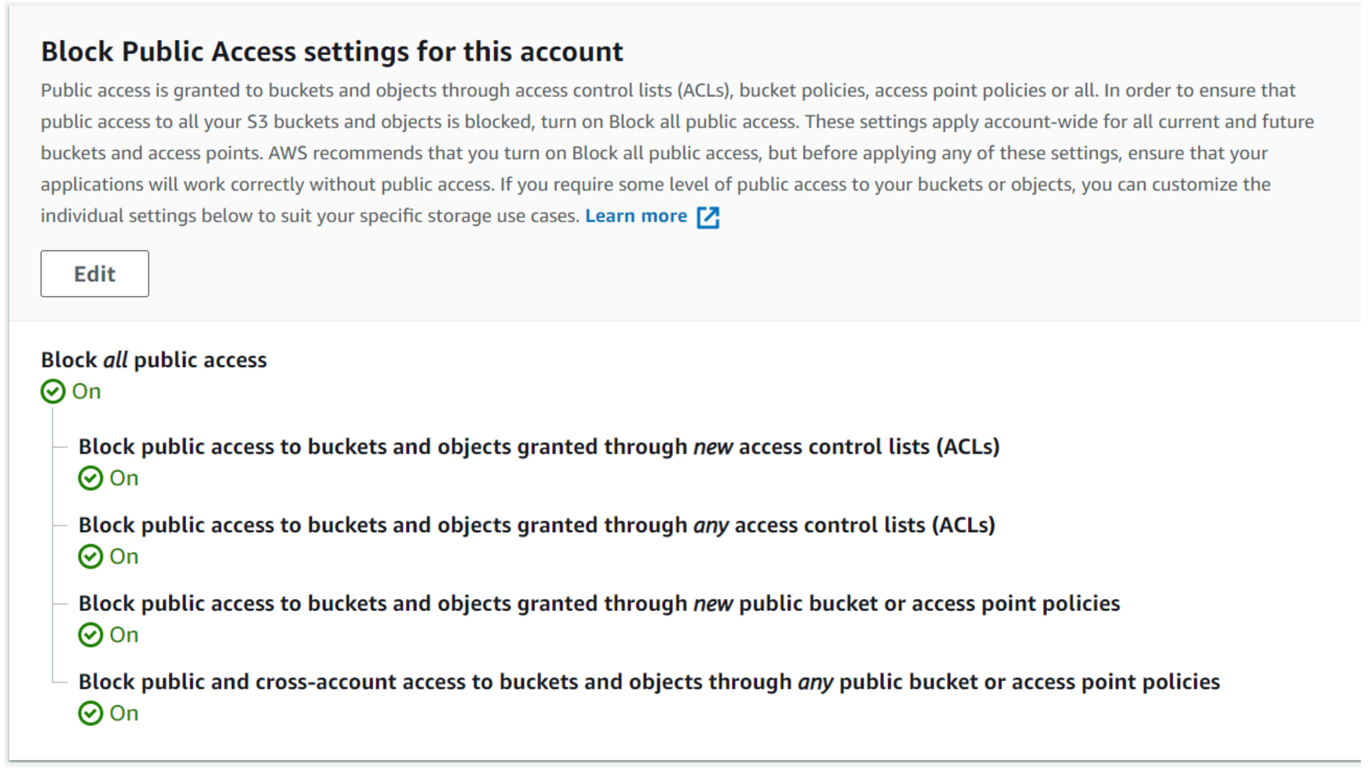
Reference: https://dataschool.com/data-modeling-101/creating-an-ec2-server/

Networking (Virtual Private Cloud):

- Virtual Private Cloud: A virtual data center with a block of IP addresses, e.g., 10.0.0.0/16 = 10.0.0.0/32 to 10.0.255.255/32 = 66536 PRIVATE IP addresses
- VPC with at least 1 public subnet with route to a Internet Gateway
- Easiest: VPC > Your VPCs > Actions > Create default VPC
- More secure: VPC > Your VPCs > Create VPC

S3 Bucket:

- S3 Block Public Access for your AWS Account
- S3 Bucket for dataset storage



Block Public Access settings for this account

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies or all. In order to ensure that public access to all your S3 buckets and objects is blocked, turn on Block all public access. These settings apply account-wide for all current and future buckets and access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to your buckets or objects, you can customize the individual settings below to suit your specific storage use cases. Learn more

Block *all* public access
✓ On

Block public access to buckets and objects granted through *new* access control lists (ACLs)
✓ On

Block public access to buckets and objects granted through *any* access control lists (ACLs)
✓ On

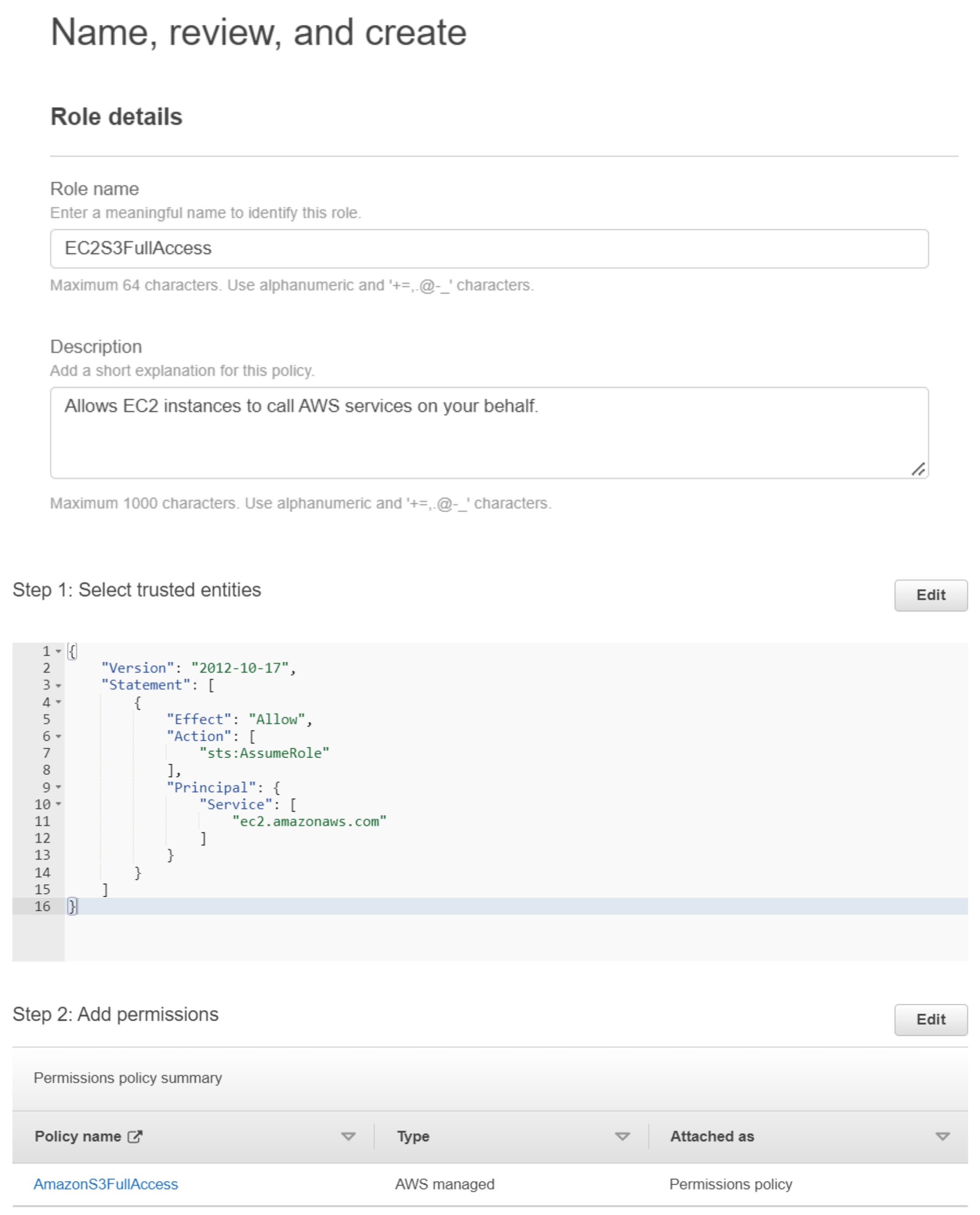
Block public access to buckets and objects granted through *new* public bucket or access point policies
✓ On

Block public and cross-account access to buckets and objects through *any* public bucket or access point policies
✓ On

Create connect to EC2 Instance (2)

IAM Role: EC2S3FullAccess

- IAM Role = Allow EC2 Instance to invoke specific AWS services
- IAM > Roles: https://console.aws.amazon.com/iamv2/home?#/roles
- Click **Create role**
    - AWS service: EC2
    - Permissions policies: AmazonS3FullAccess
    - Role name: EC2S3FullAccess
    - Review and click **Create role**


Name, review, and create

Role details

Role name
Enter a meaningful name to identify this role.
EC2S3FullAccess
Maximum 64 characters. Use alphanumeric and '+=,.@-_' characters.

Description
Add a short explanation for this policy.
Allows EC2 instances to call AWS services on your behalf.
Maximum 1000 characters. Use alphanumeric and '+=,.@-_' characters.

Step 1: Select trusted entities

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole"
            ],
            "Principal": {
                "Service": [
                    "ec2.amazonaws.com"
                ]
            }
        }
    ]
}
```

Step 2: Add permissions

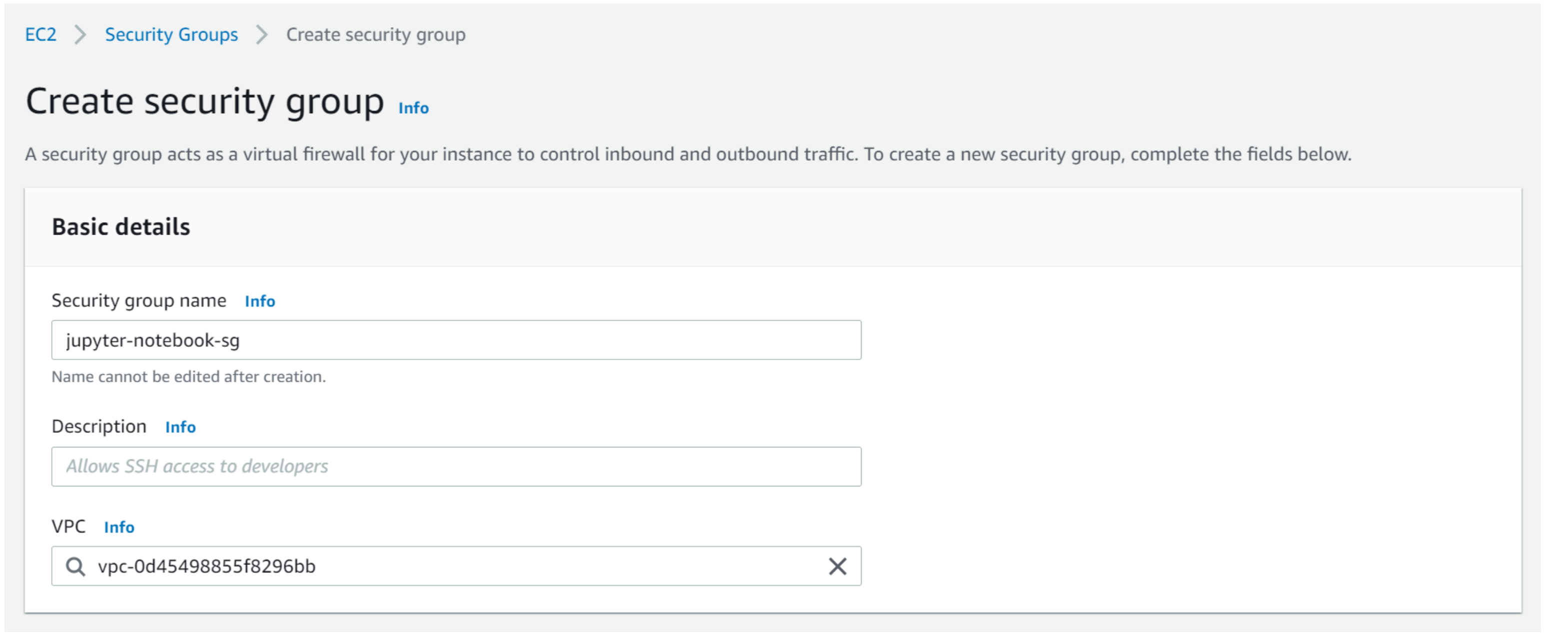
Permissions policy summary

Policy name	Type	Attached as
AmazonS3FullAccess	AWS managed	Permissions policy

Create and connect to EC2 Instance (3)

Security Group: jupyter-sg

- Security Group = Firewall for EC2 Instances
- EC2 > Security Groups: https://console.aws.amazon.com/ec2/v2/home?#SecurityGroups
- Click **Create security group**
  - Name: jupyter-sg
  - VPC: ${VPC}
  - Inbound rules
    - **SSH, 22, 0.0.0.0/0 (for demo; should be My IP)**
    - HTTPS, 443, 0.0.0.0/0
    - Custom TCP, 8888, 0.0.0.0/0



EC2 > Security Groups > Create security group

Create security group Info

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.

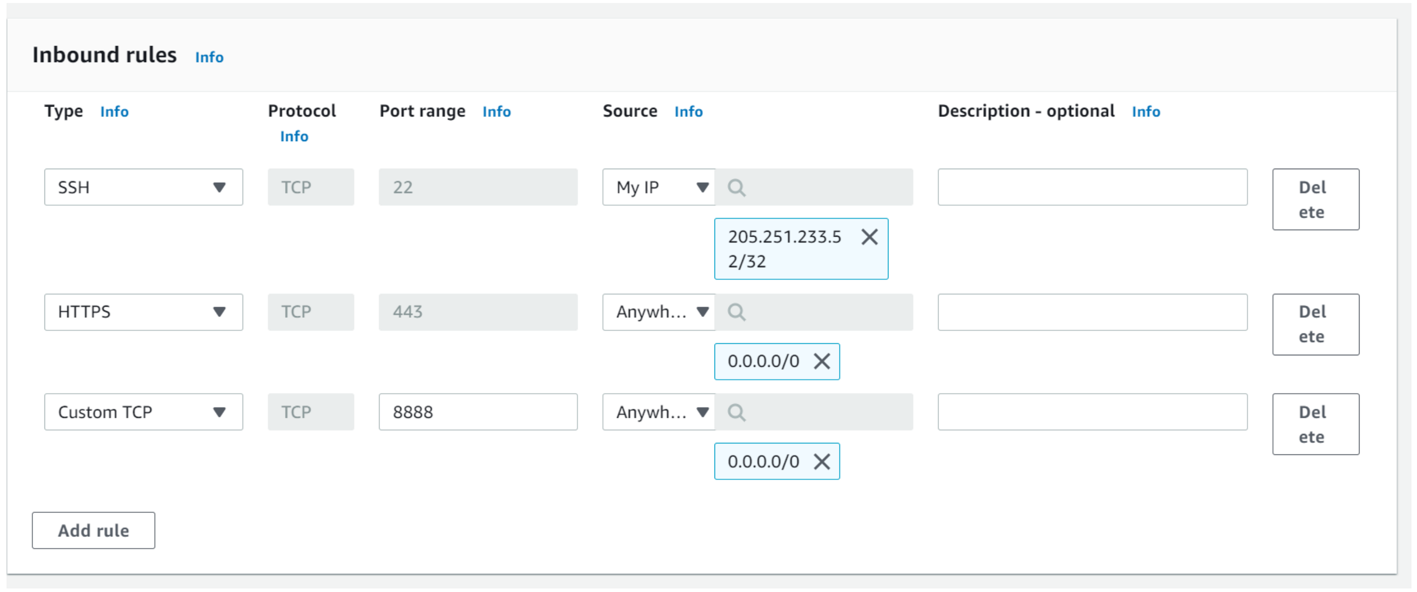
Basic details

Security group name Info: jupyter-notebook-sg

Name cannot be edited after creation.

Description Info: *Allows SSH access to developers*

VPC Info: vpc-0d45498855f8296bb



Inbound rules Info

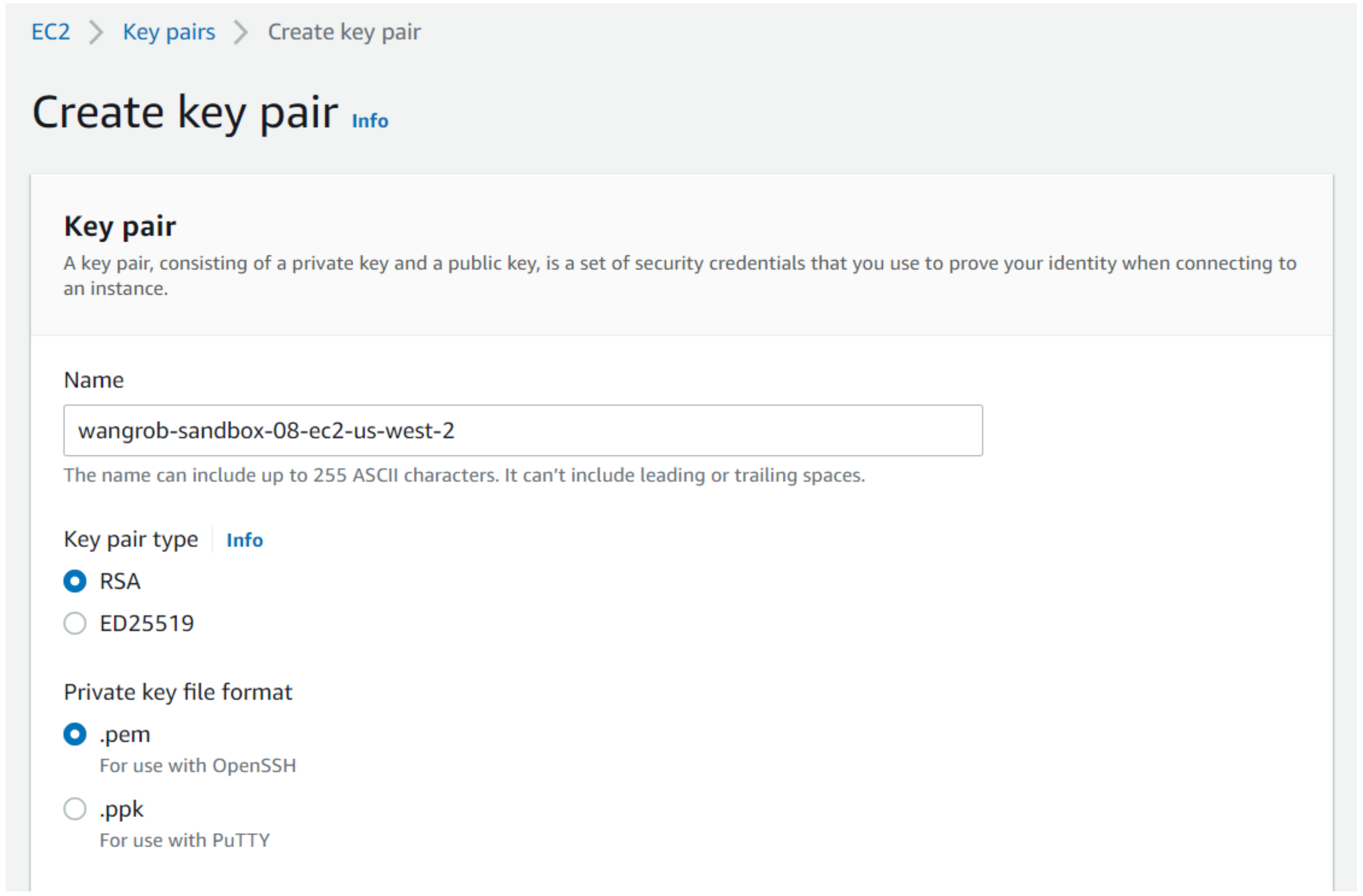
Type	Protocol	Port range	Source	Description - optional	
SSH	TCP	22	My IP 205.251.233.52/32		Delete
HTTPS	TCP	443	Anywh... 0.0.0.0/0		Delete
Custom TCP	TCP	8888	Anywh... 0.0.0.0/0		Delete

Add rule

Create and connect to EC2 Instance (4)

Key Pair: ${Account Name}-ec2-${AWS::Region}

- Key Pair = Public key on EC2 instance + Private key on SSH client
- EC2 > Key Pairs: https://console.aws.amazon.com/ec2/v2/home?#KeyPairs
- Click **Create key pair**
  - Name: ${Account Name}-ec2-${AWS::Region}
  - Key pair type: RSA (default)
  - Private key file format: .pem (for Linux OS or Git-Bash on Windows)



Key pair

A key pair, consisting of a private key and a public key, is a set of security credentials that you use to prove your identity when connecting to an instance.

Name: wangrob-sandbox-08-ec2-us-west-2

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type Info: [✓ RSA] ED25519

Private key file format: [✓ .pem For use with OpenSSH] .ppk For use with PuTTY

Create and connect to EC2 Instance (5)

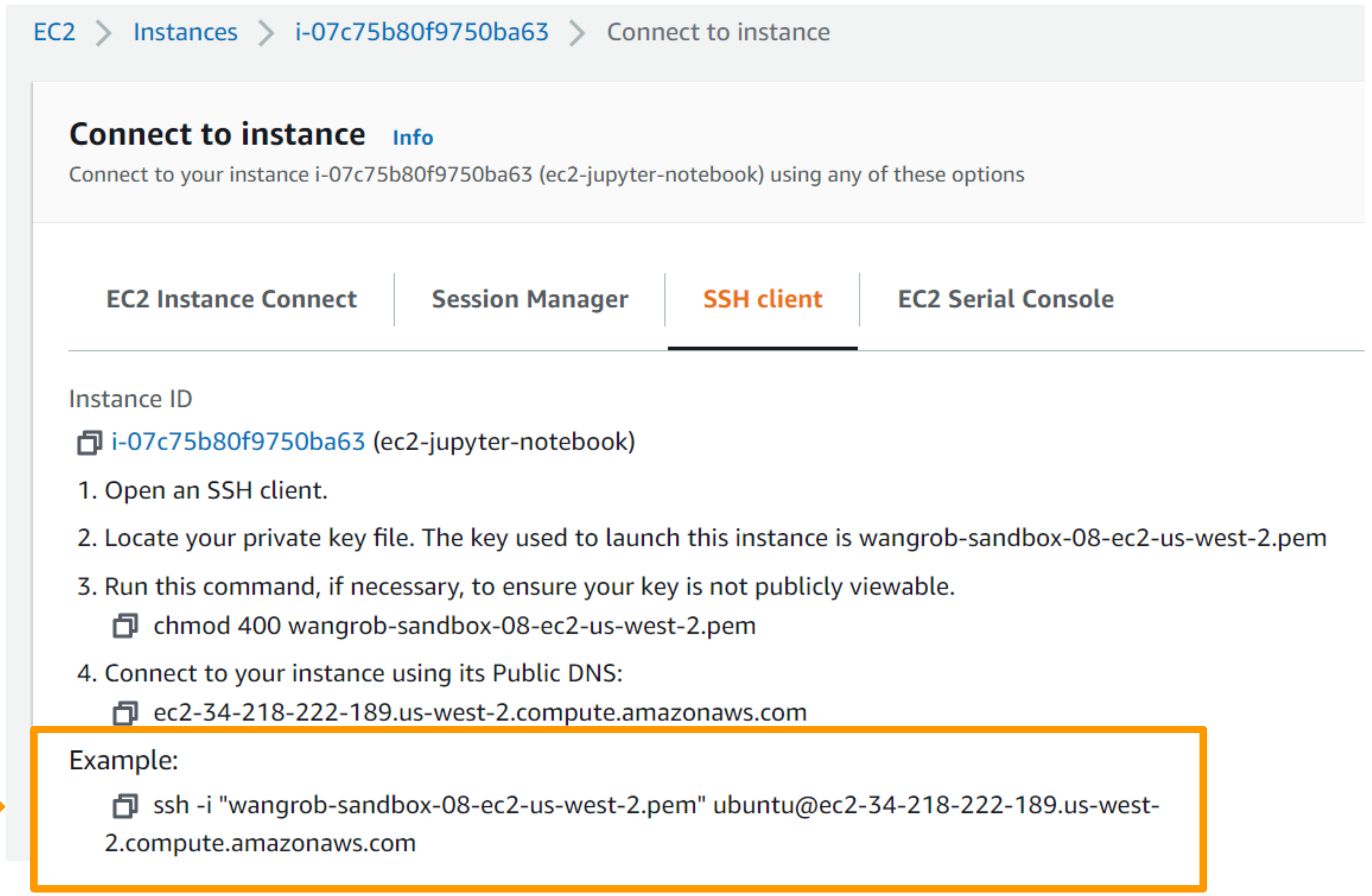
EC2 Instance: ec2-jupyter

- EC2 Instance = Virtual server
- EC2 > Instances: https://console.aws.amazon.com/ec2/v2/home?#Instances
- Click **Launch instances**
    - Name: ec2-jupyter
    - OS: Amazon Linux 2 (Free Tier eligible and prepackaged w/ AWS CLI)
    - Instance Type: m5.xlarge (CPU) or g5.xlarge (GPU)
    - Key pair: ${Key pair}
- Click **Launch instances** (cont'd)
    - Network settings > Edit
        - VPC: ${VPC}
        - Subnet: ${Public Subnet}
        - Select existing security group: jupyter-sg
    - Configure storage
        - Root volume: 10 GiB (gp3)
    - Advanced details
        - IAM instance profile: EC2S3FullAccess
    - Review and click **Launch instance**

Create and connect to EC2 Instance (6)

Connect to EC2 Instance: ec2-jupyter

- EC2 > Instances: https://console.aws.amazon.com/ec2/v2/home?#Instances
- Select ec2-jupyter
- Click **Connect** button
- Click **SSH client** tab
- Look for "Example", and copy the example command.
    - Note: Every time you restart an EC2 instance, it will get a new PUBLIC IP address (Private IP address remains the same for lifetime of the instance)



Connect to your instance i-07c75b80f9750ba63 (ec2-jupyter-notebook) using any of these options

Instance ID

i-07c75b80f9750ba63 (ec2-jupyter-notebook)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is wangrob-sandbox-08-ec2-us-west-2.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.

```
chmod 400 wangrob-sandbox-08-ec2-us-west-2.pem
```

4. Connect to your instance using its Public DNS:

```
ec2-34-218-222-189.us-west-2.compute.amazonaws.com
```

Example:

```
ssh -i "wangrob-sandbox-08-ec2-us-west-2.pem" ubuntu@ec2-34-218-222-189.us-west-2.compute.amazonaws.com
```

- Configure AWS profile
    - $ aws configure
    - Enter AWS Access Key ID and Secret Access Key

Set up Jupyter Lab Server on EC2 instance (1)

Reference: https://medium.com/@alexjsanchez/python-3-notebooks-on-aws-ec2-in-15-mostly-easy-steps-2ec5e662c6c6

Connect to EC2 instance and start w/ Step 7, 8: Download and Install Anaconda 3

- Go to Anaconda Distribution: https://www.anaconda.com/products/distribution
- Scroll down to Linux and copy link for 64-bit (x86) installer
- $ mkdir ~/installer
- $ cd ~/instller
- $ wget ${64-bit (x86) installer link}
- $ bash ${64-bit (x86) installer}.sh
- Follow and complete installation

Step 9: Set Anaconda 3 as default Python and Jupyter environment

- $ source ~/.bashrc
- $ which python
- (Confirm we are invoking python under anaconda3/bin)
- $ python --version
- $ which jupyter
- (Confirm we are invoking jupyter under anaconda3/bin)
- $ jupyter--version

Set up Jupyter Lab Server on EC2 instance (2)

Reference: https://medium.com/@alexjsanchez/python-3-notebooks-on-aws-ec2-in-15-mostly-easy-steps-2ec5e662c6c6

Download Git and test notebooks

- $ sudo yum install git –y
- $ mkdir ~/workspace
- $ cd ~/workspace
- $ git clone https://github.com/rwang5688/sm-stack-v2.git
- **Step 11: Run Jupyter Lab with --no-browser on EC2 instance**
- $ cd
- $ jupyter lab --no-browser

NOTE: Make sure Jupyter Lab is up and running BEFORE next step.

Step 10: Create dot ssh config file on client desktop

- $ nano ~/.ssh/config
- Add the following lines:

```
Host ec2-jupyter
    Hostname your-ec2's-public-ip-address here
    User ec2-user
    IdentityFile ~/${Working Directory}/${Account Name}-ec2-${AWS::Region}.pem
```

- Note: Different OS has diff. users
  - Amazon Linux 2: ec2-user
  - Ubuntu: ubuntu

Set up Jupyter Lab Server on EC2 instance (3)

Reference: https://medium.com/@alexjsanchez/python-3-notebooks-on-aws-ec2-in-15-mostly-easy-steps-2ec5e662c6c6

Step 12: Use SSH to forward local port 9999 to EC2 port 8888

- $ ssh -NfL 9999:localhost:8888 ec2-jupyter

(First time, this will ask if we want to add the EC2 Public IP to known hosts. Pls say: “yes”.)

Troubleshooting:

- If you see that Jupyter Lab on EC2 instance listening on 8889, that means another Jupyter Notebook server is running.
- $ ps –aux | grep jupyter, and then kill -9 any Jupyter process running.

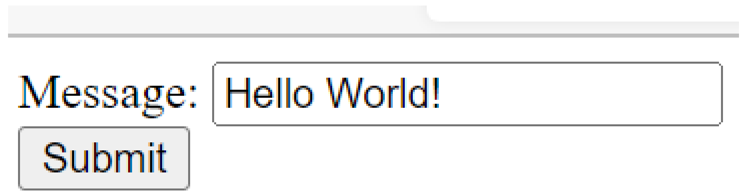
Test:

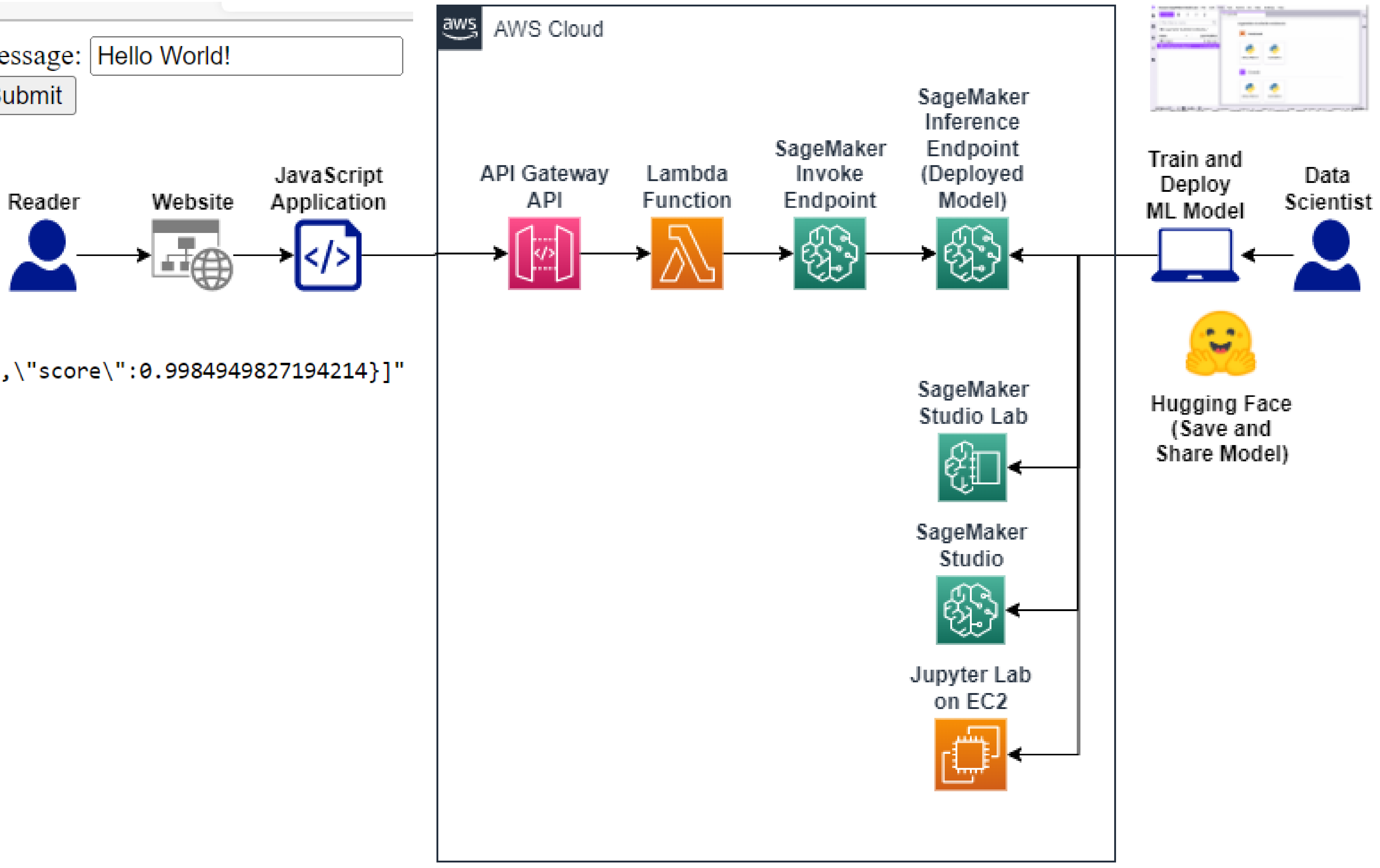
- Copy the initial HTTP request with token from Jupyter Lab Server, e.g., http://localhost:8888/?token=${TokenValue} …
- Open browser.
- Paste the initial HTTP request into the browser, change 8888 to 9999, and hit Enter.
- Find + open list_s3_buckets.ipynb, and execute the steps.

NOTE: This step assumes that you have used “aws configure” to set “default” profile.

Deploy a SageMaker-powered Content Moderation API

Text Classification Project Architecture





```
"[{\"label\":\"LABEL_1\",\"score\":0.9984949827194214}]"
```

SageMaker-powered Content Moderation API

Model deployment notebooks wraps the Hugging Face model as a **SageMaker Serverless Inference Endpoint**.

Text Classification: sst2

- PyTorch: deploy_sst2_text_classification.ipynb (SM Studio Lab or SM Studio)
- TensorFlow: deploy_sst2_text_classification-tf.ipynb (SM Studio Lab or SM Studio)

Image Classification

- Let's discuss offline ;) …

Lambda Function wraps the invocation of the SageMaker Serverless Inference Endpoint.

API Gateway API wraps the Lambda Function as a GET method.

API Gateway and Lambda combines to provide a **scalable REST API** for web applications.

Step 1: Deploy SageMaker Serverless Inference Endpoint

Reference:

Deploy A Hugging Face Pretrained Model to Amazon SageMaker Serverless Endpoint - Boto3 (Mia Chang, pymia):

https://github.com/aws/studio-lab-examples/blob/c154a9a009227c50e8af3be308fe475a7abd607f/connect-to-aws/Access_AWS_from_Studio_Lab_Deployment.ipynb

Text Classification: sst2

- PyTorch: deploy_sst2_text_classification.ipynb (SM Studio Lab or SM Studio): https://github.com/rwang5688/sm-stack-v2/blob/main/sm-model/sst2/deploy_sst2_text_classification.ipynb
- TensorFlow: deploy_sst2_text_classification-tf.ipynb (SM Studio Lab or SM Studio): https://github.com/rwang5688/sm-stack-v2/blob/main/sm-model/sst2/deploy_sst2_text_classification-tf.ipynb

Step 2: Deploy API Gateway API and Lambda Function (Prerequisites)

Install Docker

If you have:

- Linux: Install Docker
- Mac: Install Docker Desktop or Launch Cloud9
- Windows: Launch Cloud9

Verify:

- $ docker ps

Install SAM CLI (Serverless Application Model Command Line Tool)

- https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/serverless-sam-cli-install.html

Verify:

* $ sam --version

Step 2: Deploy API Gateway API and Lambda Function (1)

Edit Lambda Function

$ cd workspace

$ cd sm-stack-v2

$ cd sm-api

NOTE: Lambda function source code is under sm-api/sm-api.

Edit sm-api/get.py:

- Replace endpoint_name with your endpoint_name.
- Should be of the format: sst2-text-classification-ep-yyyy-mm-dd-hh-mm-ss.

Deploy API Gateway API and Lambda Function

$ sam build

(SAM builds the Docker container that contains the Lambda Function runtime.)

[Continue on next page.]

Step 2: Deploy API Gateway API and Lambda Function (2)

$ sam deploy --guided

Stack Name: sm-api

AWS Region: ${your region}, e.g., us-west-2, us-east-1

Confirm changes before deploy: ${your choice}, e.g., y or n

Allow SAM CLI IAM role creation: y

Disable rollback: y

GetFunction may not have authorization defined. Is this okay: y

Save arguments to a config file: y

SAM configuration file: samconfig.toml (default)

SAM configuration environment: default (default)

…

Deploy this change set: y

Wait until deployment completes.

Step 3: Configure API Gateway API (1)

Enable CORS

Navigate to API Gateway APIs console: https://console.aws.amazon.com/apigateway/main/apis …

Drill into sm-api.
Under Resources:

Select /sm-api.

Select **Actions** > **Enable CORS**.

(CORS = Cross-Origin Resource Sharing)

Gateway Response for sm-api

- Check DEFAULT 4XX and DEFAULT 5XX

Click **Enable CORS** button.

Refresh page.

Both GET and OPTIONS methods should appear.

Step 3: Configure API Gateway API (2)

Deploy API

Under Resources:

Select /sm-api.

Select **Actions** > **Deploy API**.

Deployment Stage: Prod

Click **Deploy** button.

Under Stages:

Select Prod.

Select /sm-api.

Select GET:

Save invoke URL, e.g., https://zasep4u659.execute-api.us-west-2.amazonaws.com/Prod/sm-api

Step 4: Test API Gateway API from API Gateway Console

From API Gateway API Console:

Under Resources:

Select /sm-api.

Select GET.

Click **Test**.

Under Query Strings:

Enter: message=”hello world”

Click **Test**.

(If GET returns timeout error, please wait and try again.)

Verify Response Body is:

```
"[{\"label\":\"LABEL_1\",\"score\":0.9908919930458069}]"
```

Step 5: Test API Gateway API from Web Form on Desktop

From sm-app form:

```
$ cd workspace
$ cd sm-stack-v2
$ cd sm-app
```

Edit form.html:

Replace the action URL with your Invoke URL.

E.g., https://zasep4u659.execute-api.us-west-2.amazonaws.com/Prod/sm-api

Open form.html:

Enter Message, e.g. “hello world”.

Click **Submit**.

Verify Response Body is:

```
"[{\"label\":\"LABEL_1\",\"score\":0.9908919930458069}]"
```

How can we help you innovate?