An evil-genius guide to computer programming

Daniel Lemire

Montreal 🇨🇦

blog: https://lemire.me

twitter: @lemire

GitHub: https://github.com/lemire/

Week 6

Launch It In Space

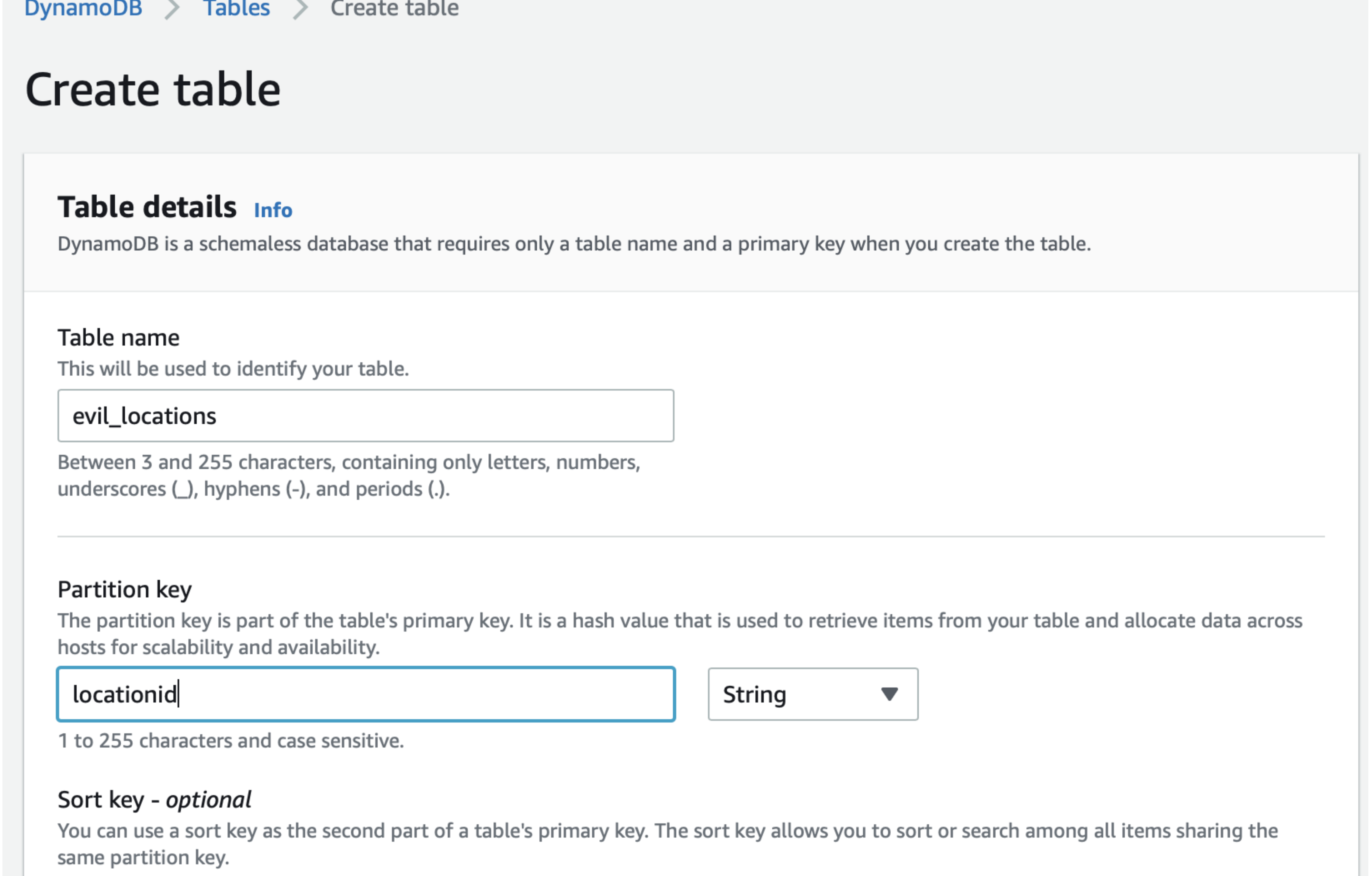
Let us build and deploy a web application

On Amazon's servers!

https://flask-service.90qakjuvfj4f8.ca-central-1.cs.amazonlightsail.com

Prerequisites

- Python
- https://www.docker.com
- https://aws.amazon.com (account)
- https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html


DynamoDB > Tables > Create table

Create table

Table details Info

DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

Table name

This will be used to identify your table.

evil_locations

Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.).

Partition key

The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.

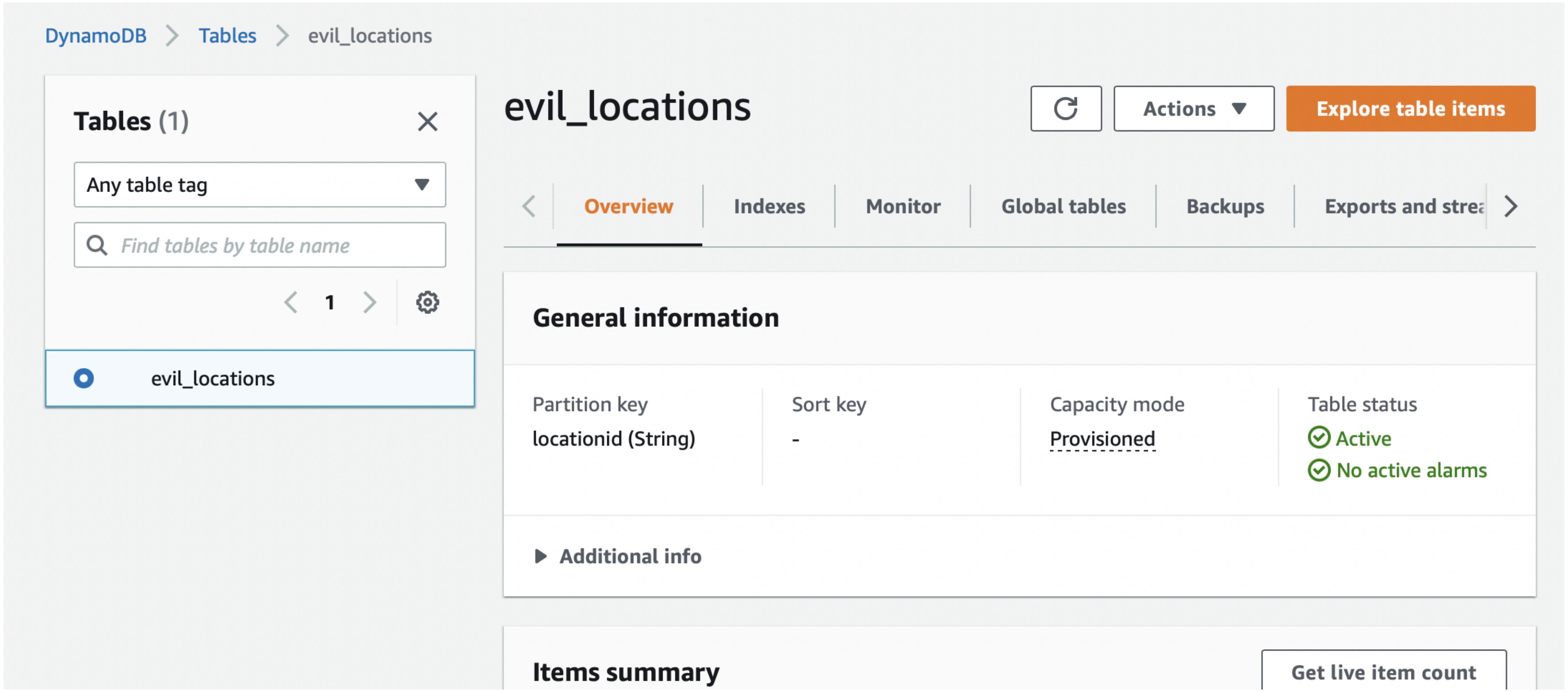
locationid

String

1 to 255 characters and case sensitive.

Sort key - *optional*

You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.



DynamoDB > Tables > evil_locations

Tables (1)

Any table tag

Find tables by table name

1

evil_locations

evil_locations

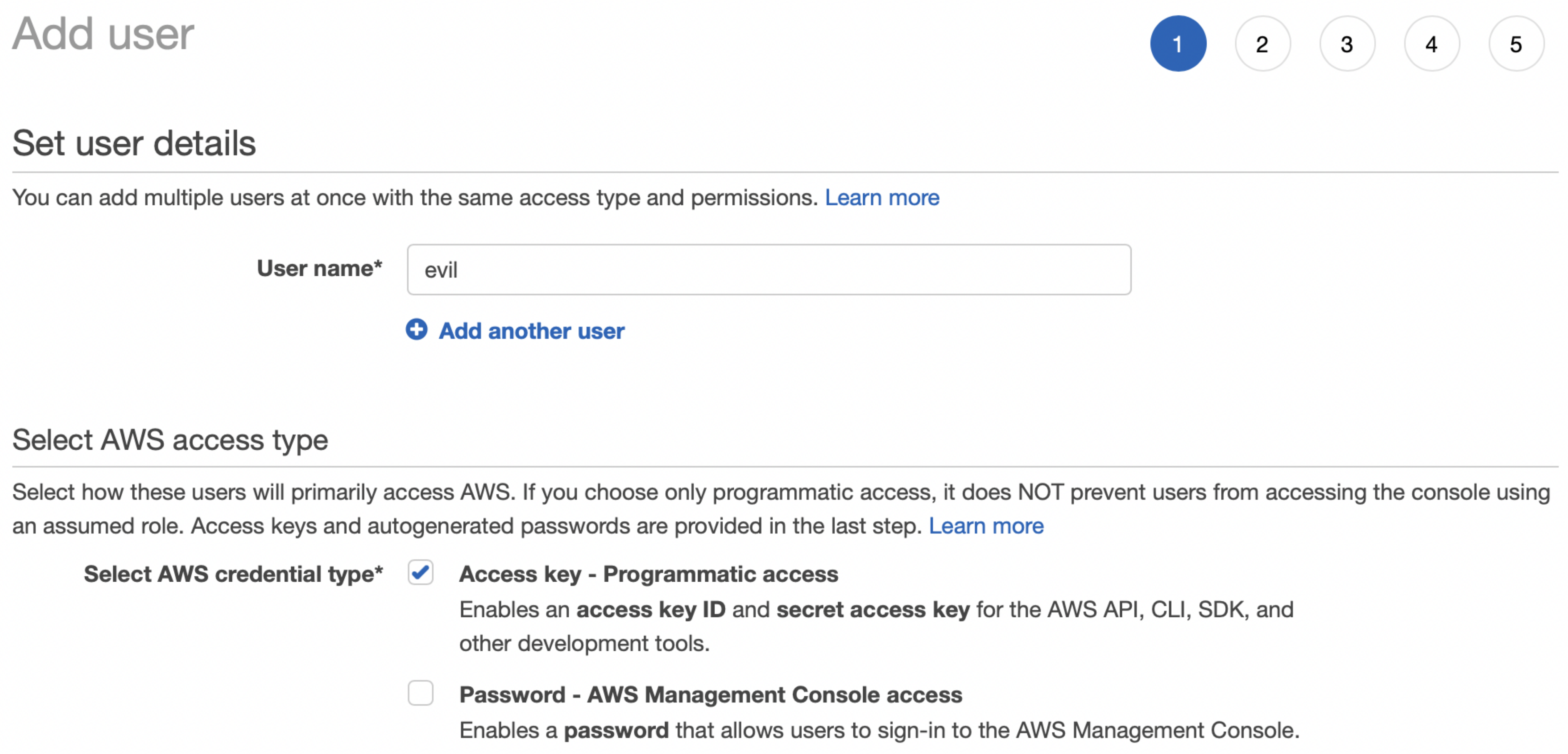
General information

Partition key	Sort key	Capacity mode	Table status
locationid (String)	-	Provisioned	Active No active alarms

▶ Additional info

Items summary

Identity and Access Management (IAM)



Add user

Set user details

You can add multiple users at once with the same access type and permissions. Learn more

User name* evil

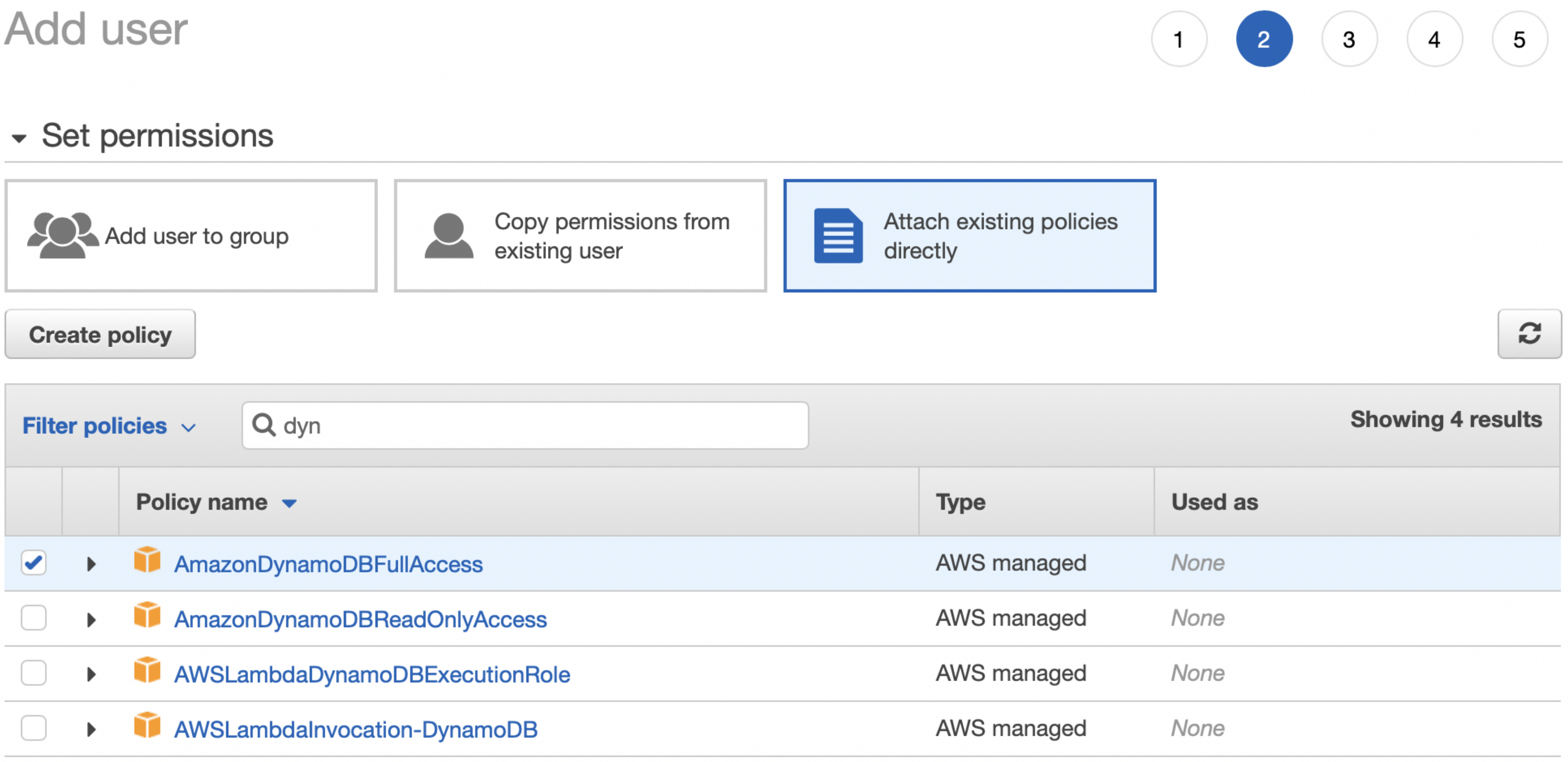
Select AWS access type

Select how these users will primarily access AWS. If you choose only programmatic access, it does NOT prevent users from accessing the console using an assumed role. Access keys and autogenerated passwords are provided in the last step. Learn more

Select AWS credential type*

[✓] **Access key - Programmatic access**
Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.

[ ] **Password - AWS Management Console access**
Enables a **password** that allows users to sign-in to the AWS Management Console.



Add user

1 2 3 4 5

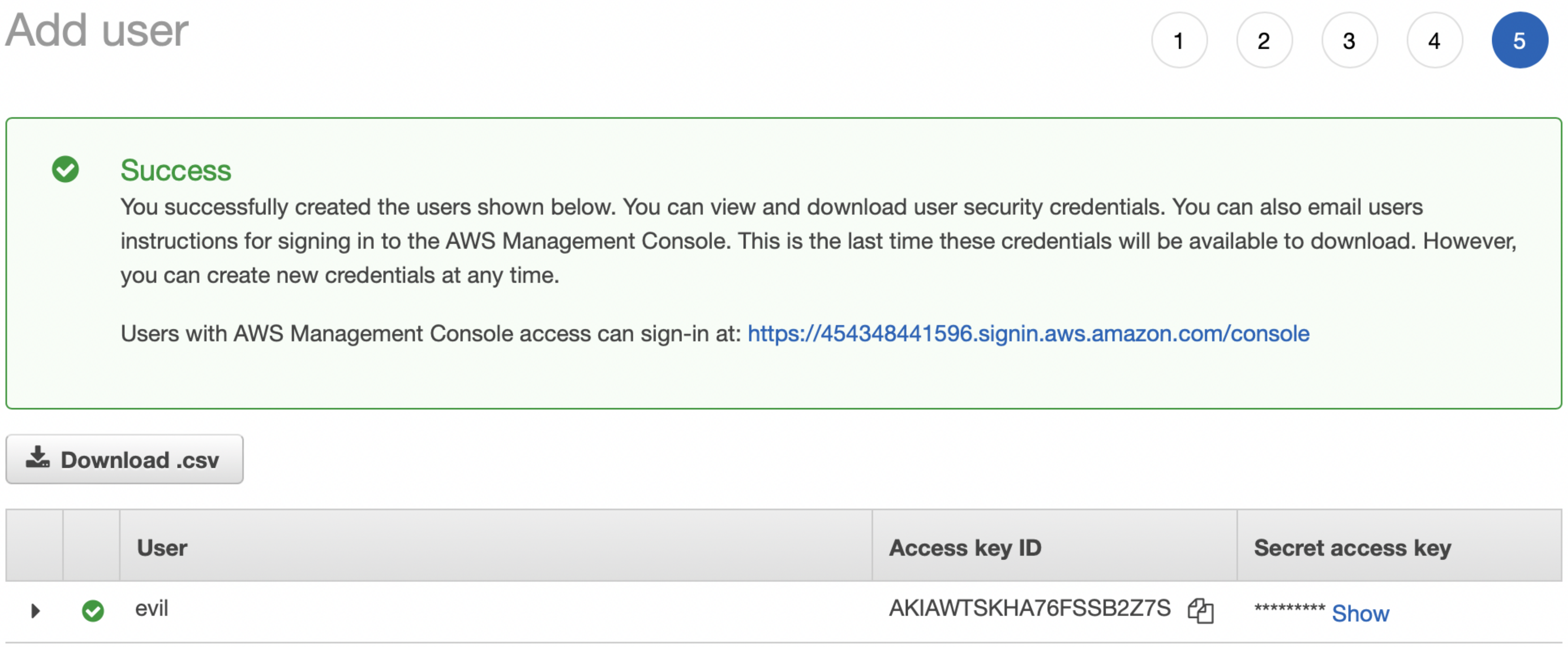
Set permissions

Add user to group
Copy permissions from existing user
Attach existing policies directly
Create policy
Filter policies

dyn

Showing 4 results

		Policy name	Type	Used as
✓	▸	AmazonDynamoDBFullAccess	AWS managed	*None*
	▸	AmazonDynamoDBReadOnlyAccess	AWS managed	*None*
	▸	AWSLambdaDynamoDBExecutionRole	AWS managed	*None*
	▸	AWSLambdaInvocation-DynamoDB	AWS managed	*None*


Add user

1 2 3 4 5

Success

You successfully created the users shown below. You can view and download user security credentials. You can also email users instructions for signing in to the AWS Management Console. This is the last time these credentials will be available to download. However, you can create new credentials at any time.

Users with AWS Management Console access can sign-in at: https://454348441596.signin.aws.amazon.com/console

Download .csv

		User	Access key ID	Secret access key
▸	✓	evil	AKIAWTSKHA76FSSB2Z7S	********* Show

```python
with open("secret.txt", "r") as file:
    secret = file.read().strip()
```

```python
import boto3
dynamodb = boto3.resource('dynamodb', aws_access_key_id="AKIAWTSKHA76FSSB2Z7S", aws_secret_access_key=secret, region_name="ca-central-1" )
```

```
>>> tables = list(dynamodb.tables.all())
>>> print("tables=",tables)
tables= [dynamodb.Table(name='evil_locations')]
```

```
>>> evil = dynamodb.Table("evil_locations")
>>> response = evil.scan()
>>> print(response)
{'Items': [{'time': '2022-05-08 17:59:31.987794', 'locationid': '75f923f6-2bdc-480e-935d-88ffa66aefa2', 'latitude': '45.54195262812087', 'longitude': '-73.48204922360495'}, ...
```

```
response = evil.scan()
print(response["Items"])
```

```
>>> import uuid
>>> myuuid = uuid.uuid4()
>>> str(myuuid)
'e30951f0-3e94-4edd-beb1-b12b2e77acc8'
```

```
response = evil.put_item(
    Item={
        'locationid': str(myuuid),
        'latitude': str(lat),
        'longitude': str(long),
    }
)
```

```html
<html>

    <body>

        <script>
            function markCoordinates() {
                if (navigator.geolocation) {
                    navigator.geolocation.getCurrentPosition(upload_to_server);
                } else {
                    x.innerHTML = "Geolocation is not supported by this browser.";
                }
            }
        </script>
    </body>
</html>
```

```html
<html>
    <head>
        <link rel="stylesheet" href="https://unpkg.com/leaflet@1.8.0/dist/leaflet.css" />
        <script src="https://unpkg.com/leaflet@1.8.0/dist/leaflet.js"></script>
    </head>

    <body>
    </body>
</html>
```

```python
app = Flask(__name__)
with open("secret.txt", "r") as file:
      secret = file.read().strip()
```

```python
@app.route('/')
def upload_file_render():
    return render_template('map.html')
```

```
<html>
        <script>

            function retrieveSaved(draw) {
               fetch("/query")
               .then(res => res.json())
               .then(out =>
                 { draw(out["Items"]);})
            }
        </script>
    </body>
</html>
```

```
@app.route('/query')
def query_function():
    dynamodb = boto3.resource('dynamodb', aws_access_key_id="AKIAWTSKHA76FSSB2Z7S", aws_secret_access_key=secret, region_name="ca-central-1" )
    evil = dynamodb.Table("evil_locations")
    return jsonify(evil.scan())
```

```
<html>
        <script>

            function upload_to_server(position) {
                fetch("/add?lat="+position.coords.latitude+"&long="+position.coords.longitude)
               .then(out => { update(position); }
               )
            }
        </script>
    </body>
</html>
```

```python
@app.route('/add')
def add_function():
    dynamodb = boto3.resource('dynamodb', aws_access_key_id="AKIAWTSKHA76FSSB2Z7S", aws_secret_access_key=secret, region_name="ca-central-1" )
    evil = dynamodb.Table("evil_locations")
    args = request.args
    lat=args["lat"]
    long=args["long"]
    dt = str(datetime.now())
    return jsonify(evil.put_item(
              Item = {
               'locationid': str(uuid.uuid4()),
               'latitude': str(lat),
               'longitude': str(long),
               'time': dt
             }
            ))
```

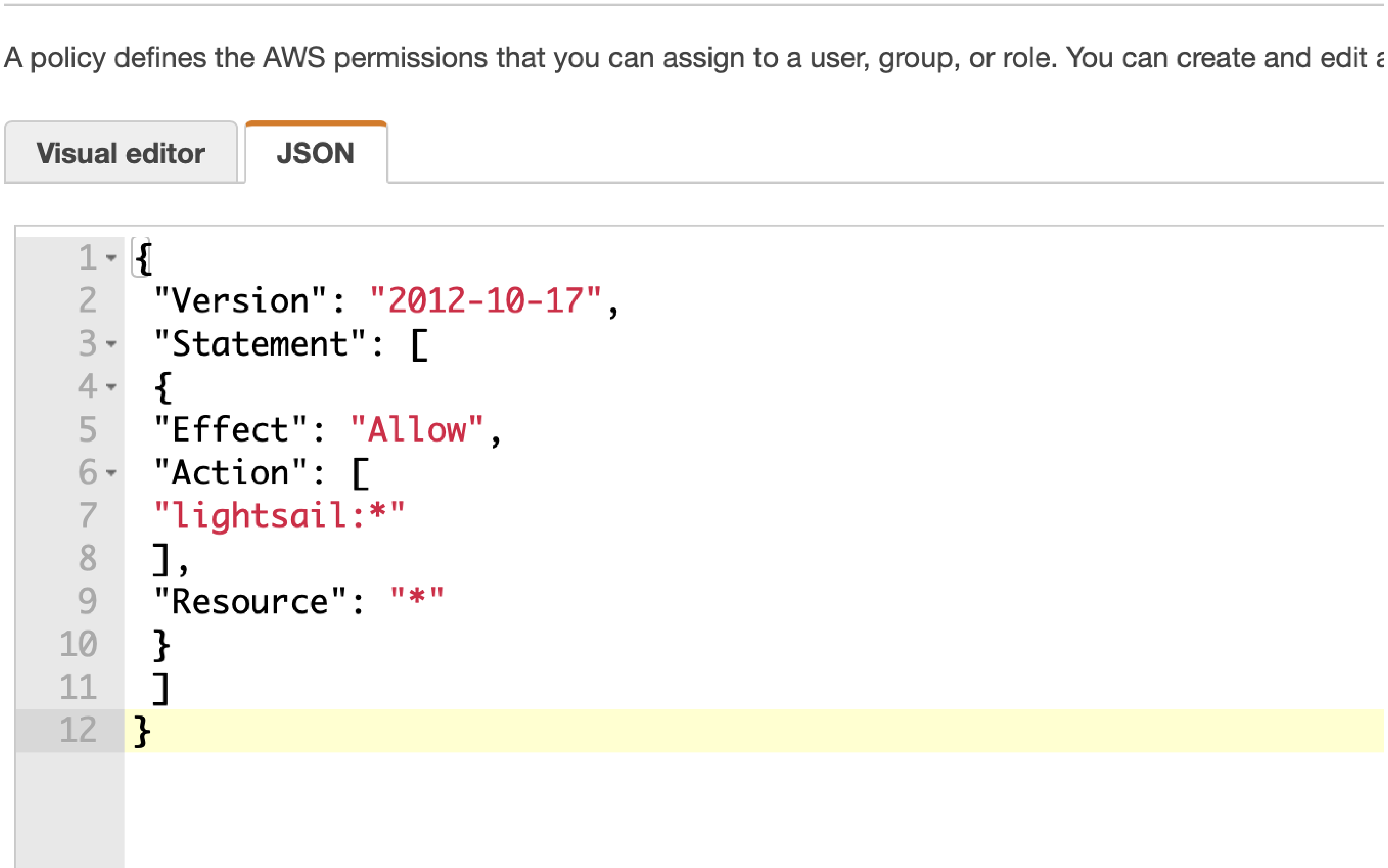
requirements.txt

```
boto3==1.22.9
Flask==2.1.1
importlib-metadata==4.11.3
plum-py==0.7.9
python-dateutil==2.8.2
```

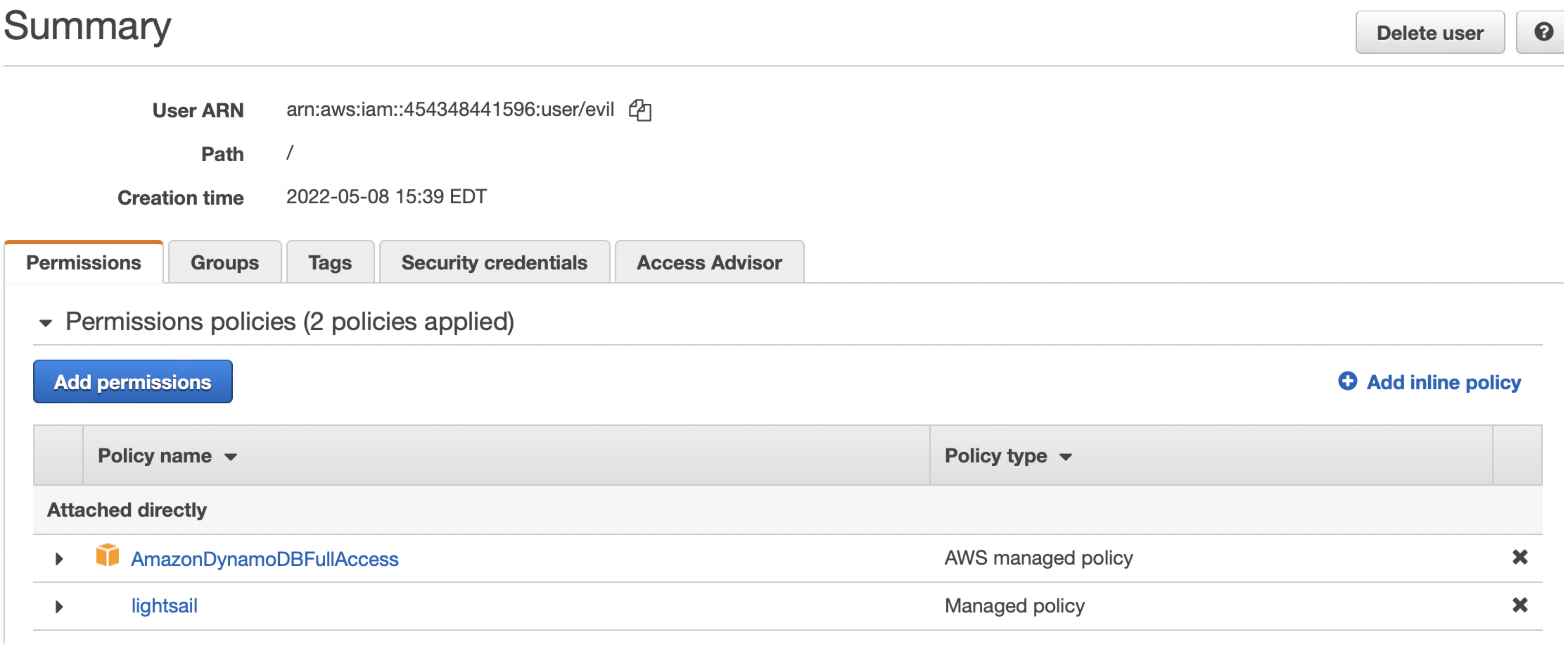
Dockerfile

```
# Set base image (host OS)
FROM python:3.8-alpine

# By default, listen on port 5005
EXPOSE 5005/tcp

# Set the working directory in the container
WORKDIR /app

# Copy the dependencies file to the working directory
COPY requirements.txt .

# Install any dependencies
RUN pip install -r requirements.txt

# Copy the content of the local src directory to the working directory
COPY server.py .
RUN mkdir templates
RUN mkdir static
COPY templates/ ./templates
COPY static/ ./static
COPY secret.txt .

# Specify the command to run on container start
CMD [ "python", "./server.py" ]
```

https://lightsail.aws.amazon.com/ls/docs/en_us/articles/amazon-lightsail-install-software https://aws.amazon.com/getting-started/hands-on/serve-a-flask-app/


A policy defines the AWS permissions that you can assign to a user, group, or role. You can create and edit a

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "lightsail:*"
 ],
 "Resource": "*"
 }
 ]
}
```



Summary

Delete user

User ARN	arn:aws:iam::454348441596:user/evil
Path	/
Creation time	2022-05-08 15:39 EDT

Permissions policies (2 policies applied)

Policy name	Policy type
Attached directly	
AmazonDynamoDBFullAccess	AWS managed policy
lightsail	Managed policy

```
docker build -t flask-container .

docker buildx build --platform=linux/amd64 -t flask-container .
docker image rm -f flask-container
docker run -p 5005:5005 flask-container

http://127.0.0.1:5005
```

```
❯  aws configure
❯ aws lightsail create-container-service --service-name flask-service  --power small --scale 1
❯ aws lightsail create-container-service --service-name flask-service  --power small --scale 1
{
    "containerService": {
        "containerServiceName": "flask-service",
        "arn": "arn:aws:lightsail:ca-central-1:454348441596:ContainerService/5a30d679-3492-4fa7-9f76-c393ee207515",
        "createdAt": "2022-05-10T21:44:26-04:00",
        "location": {
            "availabilityZone": "all",
            "regionName": "ca-central-1"
        },
        "resourceType": "ContainerService",
        "tags": [],
        "power": "small",
        "powerId": "small-1",
        "state": "PENDING",
        "scale": 1,
        "isDisabled": false,
        "principalArn": "",
        "privateDomainName": "flask-service.service.local",
        "url": "https://flask-service.90qakjuvfj4f8.ca-central-1.cs.amazonlightsail.com/"
    }
}
```

```
❯ aws lightsail push-container-image --service-name flask-service --label flask-container --image flask-container
8e2315ef24aa: Pushed
14780e456137: Pushed
89b59c4bc52f: Pushed
6ff9327bb7d0: Pushed
4ec9431c6304: Pushed
5efa6ede5a9a: Pushed
7f57f0a5ddf1: Pushed
d3fcb69604f8: Pushed
b53145450765: Pushed
f515aa8aa37d: Pushed
a8b3bd8be6fa: Pushed
96c8d77b0ba8: Pushed
3618203fbcc2: Pushed
4f4ce317c6bb: Pushed
Digest: sha256:e6258839c29b2567db1d1194a3cc7176c0871d2a086185c85627e9edf8938641
Image "flask-container" registered.
Refer to this image as ":flask-service.flask-container.1" in deployments.
```

```
❯ more containers.json
{
    "flask": {
        "image": ":flask-service.flask-container.1",
        "ports": {
            "5005": "HTTP"
        }
    }
}
```

```
❯ more public-endpoint.json
{
    "containerName": "flask",
    "containerPort": 5005
}
```

```
❯ aws lightsail create-container-service-deployment  --service-name flask-service --containers file://containers.json --public-endpoint file://public-endpoint.json
{
    "containerService": {
        "containerServiceName": "flask-service",
        "arn": "arn:aws:lightsail:ca-central-1:454348441596:ContainerService/5a30d679-3492-4fa7-9f76-c393ee207515",
        "createdAt": "2022-05-10T21:44:26-04:00",
        "location": {
            "availabilityZone": "all",
            "regionName": "ca-central-1"
        },
        "resourceType": "ContainerService",
        "tags": [],
        "power": "small",
        "powerId": "small-1",
        "state": "DEPLOYING",
        "scale": 1,
        "nextDeployment": {
            "version": 2,
            "state": "ACTIVATING",
            "containers": {
                "flask": {
                    "image": ":flask-service.flask-container.2",
                    "command": [],
                    "environment": {},
                    "ports": {
                        "5005": "HTTP"
                    }
                }
            },
            "publicEndpoint": {
                "containerName": "flask",
                "containerPort": 5005,
                "healthCheck": {
                    "healthyThreshold": 2,
                    "unhealthyThreshold": 2,
                    "timeoutSeconds": 2,
                    "intervalSeconds": 5,
                    "path": "/",
                    "successCodes": "200-499"
                }
            },
            "createdAt": "2022-05-10T22:11:31-04:00"
        },
        "isDisabled": false,
        "principalArn": "arn:aws:iam::170441448102:role/amazon/lightsail/ca-central-1/containers/flask-service/38e4eeivug83dqg0rh18cql1sp6jkd47lutt3k3c8ael6ebvh31g",
        "privateDomainName": "flask-service.service.local",
        "url": "https://flask-service.90qakjuvfj4f8.ca-central-1.cs.amazonlightsail.com/"
    }
}
```

```
❯ aws lightsail get-container-services --service-name flask-service
{
    "containerServices": [
        {
            "containerServiceName": "flask-service",
            "arn": "arn:aws:lightsail:ca-central-1:454348441596:ContainerService/5a30d679-3492-4fa7-9f76-c393ee207515",
            "createdAt": "2022-05-10T21:44:26-04:00",
            "location": {
                "availabilityZone": "all",
                "regionName": "ca-central-1"
            },
            "resourceType": "ContainerService",
            "tags": [],
            "power": "small",
            "powerId": "small-1",
            "state": "DEPLOYING",
❯ aws lightsail get-container-services --service-name flask-service
{
    "containerServices": [
        {
            "containerServiceName": "flask-service",
            "arn": "arn:aws:lightsail:ca-central-1:454348441596:ContainerService/5a30d679-3492-4fa7-9f76-c393ee207515",
            "createdAt": "2022-05-10T21:44:26-04:00",
            "location": {
                "availabilityZone": "all",
                "regionName": "ca-central-1"
            },
            "resourceType": "ContainerService",
            "tags": [],
            "power": "small",
            "powerId": "small-1",
            "state": "READY",
            "scale": 1,
            "isDisabled": false,
            "principalArn": "arn:aws:iam::170441448102:role/amazon/lightsail/ca-central-1/containers/flask-service/38e4eeivug83dqg0rh18cql1sp6jkd47lutt3k3c8ael6ebvh31g",
            "privateDomainName": "flask-service.service.local",
            "url": "https://flask-service.90qakjuvfj4f8.ca-central-1.cs.amazonlightsail.com/"
        }
    ]
}
```

```
aws lightsail delete-container-service --service-name flask-service
```

https://lightsail.aws.amazon.com/ls/docs/en_us/articles/amazon-lightsail-point-domain-to-distribution https://console.aws.amazon.com/billing/