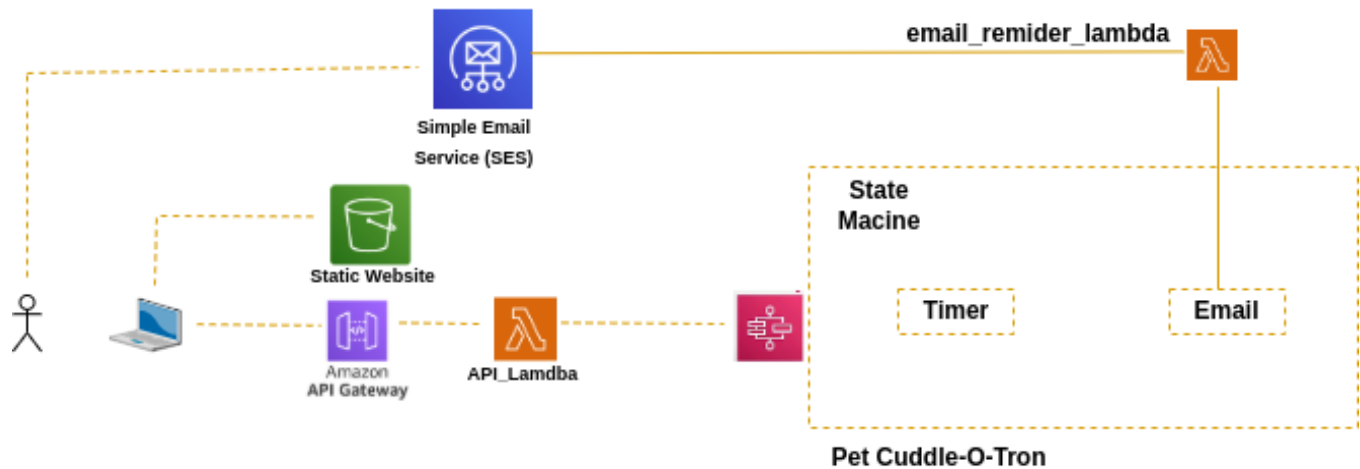


"Pet Cuddle-O-Tron" allows users to receive email reminders about events related to their pets using AWS services:



● STAGE 1 : Configure Simple Email service:

Stage 1 - Verifying Email Addresses for SES

The **Pet-Cuddle-O-Tron** application will send reminder messages via **SMS and Email** using **AWS Simple Email Service (SES)**. In a production environment, SES can be configured to send emails to any user of the application.

By default, SES operates in **sandbox mode**, meaning it can only send emails to **verified addresses** to prevent spam.

For this demo, instead of removing SES from sandbox mode, we will **verify the sender and receiver email addresses** to keep things simple.

Stage 1A - Verify the Application's Sending Email Address

Steps to verify the sender email address:

1. Ensure you are logged into an AWS account with **admin privileges** and are in the **us-east-1 (N. Virginia) region**.
2. Open the **SES Console**:
👉 [AWS SES Console](#)
3. Click on **Verified Identities** under **Configuration**.
4. Click **Create Identity**.
5. Select the **Email Address** option.
6. Enter an email address that the application will use to send emails.
 - Example: `adrian+cuddleotron@cantrill.io`

- You can use any valid email address, but it must be one you can access.
 - 7. Click **Create Identity**.
 - 8. Check your inbox for a **verification email** and click the confirmation link inside.
 - 9. You should see a **Congratulations!** message.
 - 10. Return to the **SES Console** and refresh the page. The email address should now be marked as **Verified**.
 - 11. **Save this email address**, as it will be used later as the **PetCuddleOTron Sending Address**.
-

Stage 1B - Verify the Customer Email Address

If you want to use a different email address for testing, follow these steps:

1. Go back to the **SES Console**.
 2. Click **Create Identity**.
 3. Select the **Email Address** option.
 4. Enter an email for the test customer (for example, `adrian+cuddlecustomer@cantrill.io`).
 5. Click **Create Identity**.
 6. Check your inbox for a **verification email** and click the confirmation link inside.
 7. You should see a **Congratulations!** message.
 8. Return to the **SES Console** and refresh the page. The email should now be marked as **Verified**.
 9. **Save this email address** as the **PetCuddleOTron Customer Address**.
-

Stage 1 - Completion

At this point, you have successfully verified **two email addresses** for use with SES:

- ✓ The **PetCuddleOTron Sending Address** (used by the application to send emails).
- ✓ The **PetCuddleOTron Customer Address** (used to receive test emails).

These email addresses will be configured and used in later stages of the application setup.

You have now completed all the tasks for Stage 1 of this Advanced Demo Lesson! 🎉

1 lists your domains, subdomains, and email address identities. All identities must be verified before you use them to send email in Amazon SES. [Learn more](#) 2. The **Recommendations** pane lists high-impact email authentication identities you select and check for recommendations. [Learn more](#)

2) Info

Check for recommendations Send test email Delete Create identity

Identities

< 1 > ⚙

Identity type	Identity status
Email address	Verified
Email address	Verified

Recommendations (0) Info

⚙

Identities

Recommendations

Identity name	Age	Recommendation/Description	Last checked	Resolve issue
No recommendations found				

Select up to 10 verified domain identities and click **Check for recommendations** on the Identities table above.

Enable Virtual Deliverability Manager to generate recommendations automatically

● STAGE 2 : Add a email lambda function to use SES to send emails for the serverless application

In this stage, we will **create an IAM role** and a **Lambda function** that will handle sending emails using **AWS SES**.

Stage 2A - Create the IAM Role for Lambda

The `email_reminder_lambda` function needs permissions to interact with other AWS services like **SES**, **SNS**, and **CloudWatch Logs**. Instead of manually creating an IAM role, we will use **AWS CloudFormation** to automate the process.

Steps to create the IAM role:

1. Click this link to open **CloudFormation**:
👉 [CloudFormation Quick Create](#)
2. Check the box **"I acknowledge that AWS CloudFormation might create IAM resources."**
3. Click **Create Stack**.
4. Wait until the stack reaches the **CREATE_COMPLETE** state.

5. Open the **IAM Console**:

👉 [IAM Roles](#)

6. Find the new IAM role and review its permissions.

- It allows access to **SES, SNS, and logging** services.
- This role will be used by the Lambda function to interact with AWS services.

The screenshot shows the AWS CloudFormation console. The left sidebar contains navigation links for CloudFormation, Stacks, Stack details, Drifts, StackSets, and Exports. The main content area displays a list of stacks under the heading "Stacks (7)". The stacks are listed in a table with columns for Stack name, Status, Created time, and Description. The first stack, "StateMachineRole", is highlighted with a blue selection bar. The status for all stacks is "CREATE_COMPLETE".

Stack name	Status	Created time	Description
StateMachineRole	CREATE_COMPLETE	2025-03-01 21:30:28 UTC-0400	-
LAMBDA_ROLE	CREATE_COMPLETE	2025-03-01 21:19:02 UTC-0400	-
StackSet-AWSControlTowerBP-BASELINE-CONFIG-26988fa4-4f1f-4ded-8bf7-73f5e1508f2e	CREATE_COMPLETE	2023-11-11 17:54:39 UTC-0400	Configure AWS Config
StackSet-AWSControlTowerBP-BASELINE-CLOUDWATCH-69b0e5e4-6488-4291-a0ea-44ea3796ed0	CREATE_COMPLETE	2023-11-11 17:54:39 UTC-0400	Configure Cloudwatch Rule, local SNS Topic, forwarding notifications from local SNS Topic to Security Topic
StackSet-AWSControlTowerBP-BASELINE-ROLES-7d230476-7077-4613-9d95-22af4d79fe3	CREATE_COMPLETE	2023-11-11 17:53:08 UTC-0400	Configure the Cross-Account IAM Security Roles for the member accounts.
StackSet-AWSControlTowerBP-BASELINE-SERVICE-ROLES-bf4c2cb0-12ed-404b-a5f7-6658b3597f42	CREATE_COMPLETE	2023-11-11 17:53:08 UTC-0400	Configure AWS Config and SNS Notification Forward IAM Roles
StackSet-AWSControlTowerBP-BASELINE-SERVICE-LINKED-ROLE-d41e7931-b342-4410-88bc-752d51105c0e	CREATE_COMPLETE	2023-11-11 17:53:07 UTC-0400	Configure Control Tower Service Linked Role


The screenshot shows the AWS IAM console. The left sidebar contains navigation links for Identity and Access Management (IAM), Dashboard, Access management, Users, Roles, Policies, Identity providers, Account settings, Root access management, Access reports, Access Analyzer, External access, Unused access, Analyzer settings, Credential report, Organization activity, Service control policies, Resource control policies, IAM Identity Center, and AWS Organizations. The main content area displays a list of roles under the heading "Roles (33)". The roles are listed in a table with columns for Role name, Trusted entities, and Last activity. The first role, "aws-controltower-AdministratorExecutionRole", is highlighted with a blue selection bar. The last activity for the first five roles is shown as "11 days ago", "99 days ago", and "10 hours ago".

Role name	Trusted entities	Last activity
aws-controltower-AdministratorExecutionRole	Account: 495548951594	-
aws-controltower-ConfigRecorderRole	AWS Service: config	-
aws-controltower-ForwardSnsNotificationRole	AWS Service: lambda	11 days ago
aws-controltower-ReadOnlyExecutionRole	Account: 495548951594	-
AWSControlTowerExecution	Account: 381134089987	99 days ago
AWSReservedSSO_AWSAdministratorAccess_67cab2fdc14efcd1	Identity Provider: amawsiam:4317	-
AWSReservedSSO_AWSOrganizationsFullAccess_bf14f9e36007fd0	Identity Provider: amawsiam:4317	-
AWSReservedSSO_AWSPowerUserAccess_cbdbadd79958460	Identity Provider: amawsiam:4317	-
AWSReservedSSO_AWSReadOnlyAccess_9bd85981b659abf3	Identity Provider: amawsiam:4317	-
AWSServiceRoleForAmazonEKS	AWS Service: eks (Service-Linked Role)	10 hours ago
AWSServiceRoleForAmazonEKSNodegroup	AWS Service: eks-nodegroup (Service-Linked Role)	10 hours ago
AWSServiceRoleForAmazonGuardDuty	AWS Service: guardduty (Service-Linked Role)	16 minutes ago
AWSServiceRoleForAmazonGuardDutyMalwareProtection	AWS Service: malware-protection.gu.	-
AWSServiceRoleForAmazonSSM	AWS Service: ssm (Service-Linked Role)	56 minutes ago
AWSServiceRoleForAPIGateway	AWS Service: ops.apigateway (Service-Linked Role)	-
AWSServiceRoleForAutoScaling	AWS Service: autoscaling (Service-Linked Role)	10 hours ago
AWSServiceRoleForAWSControlTower	AWS Service: controltower (Service-Linked Role)	465 days ago

Stage 2B - Create the Lambda Function

Next, we will create the **email_reminder_lambda** function. This function will generate and send emails through **AWS SES** when triggered.

Steps to create the Lambda function:

1. Open the **AWS Lambda Console**:
 [Lambda Console](#)
 2. Click **Create Function**.
 3. Choose **Author from scratch**.
 4. Enter the function name: **email_reminder_lambda**.
 5. Under **Runtime**, select **Python 3.9**.
 6. Expand **Change default execution role**.
 7. Select **Use an existing role**.
 8. Choose the IAM role you created earlier (**LambdaRole**).
 9. Click **Create Function**.
-

Stage 2C - Configure the Lambda Function

Now, we need to add code to the Lambda function so it can send emails.

Steps to configure the function:

1. Scroll down to **Function Code**.
2. Delete the existing code and replace it with this:

python

CopyEdit

```
import boto3, os, json
```

```
FROM_EMAIL_ADDRESS = 'REPLACE_ME'
```

```
ses = boto3.client('ses')
```

```
def lambda_handler(event, context):
```

```
    # Print event data to logs
```

```
    print("Received event: " + json.dumps(event))
```

```
    # Send an email using SES
```

```
    ses.send_email(
```

```

        Source=FROM_EMAIL_ADDRESS,
        Destination={'ToAddresses':
[event['Input']['email']]},
        Message={
            'Subject': {'Data': 'Whiskers Commands You to
Attend!'},
            'Body': {'Text': {'Data':
event['Input']['message']}}
        }
    )
    return 'Success!'

```

3. Replace **'REPLACE_ME'** with the **PetCuddleOTron Sending Address** you verified in **Stage 1**.
4. Click **Deploy** to save the function.
5. Scroll to the top and copy the **Lambda function ARN** (Amazon Resource Name).
6. Save this ARN for later use as the **email_reminder_lambda ARN**.

Stage 2 - Completion

At this point, you have successfully:

- ✅ Created an **IAM Role** for the Lambda function with SES, SNS, and logging permissions.
- ✅ Created the **email_reminder_lambda** function.
- ✅ Configured the function to send emails using **AWS SES**.

You are now ready to proceed to **Stage 3** of the demo! 🚀

- **STAGE 3 : Implement and configure the state machine, the core of the application**

In this stage, we will **create an IAM role** for the state machine and then configure the **AWS Step Functions state machine** to manage the workflow of our serverless application.

Stage 3A - Create the IAM Role for the State Machine

The state machine needs permissions to interact with **AWS services** such as **CloudWatch Logs, Lambda (for sending emails), and SNS (for sending text messages)**. Instead of creating this role manually, we will use **AWS CloudFormation** to automate the process.

Steps to create the IAM role:

1. Click this link to open **CloudFormation**:
👉 [CloudFormation Quick Create](#)
 2. Check the box "**I acknowledge that AWS CloudFormation might create IAM resources.**"
 3. Click **Create Stack**.
 4. Wait for the stack to reach the **CREATE_COMPLETE** state.
 5. Open the **IAM Console**:
👉 [IAM Roles](#)
 6. Find the newly created **State Machine Role** and review its permissions:
 - It allows **logging** to CloudWatch.
 - It has permission to **invoke the email Lambda function**.
 - It can **send SMS messages via SNS**.
-

Stage 3B - Create the State Machine

Now, we will create the **state machine**, which will define the workflow for the serverless application.

Steps to create the state machine:

1. Open the **AWS Step Functions Console**:
👉 [Step Functions Console](#)
2. Click the **menu (≡)** in the **top left** and select **State Machines**.
3. Click **Create State Machine**.
4. Select **"Write your workflow in code"** to use **Amazon States Language (ASL)**.
5. Scroll down to **Type** and select **Standard**.
6. Open this link in a new tab:
👉 [Pet Cuddle-O-Tron State Machine JSON](#)

7. Copy all the JSON content to your clipboard.
8. Go back to the **Step Functions Console**.
9. Delete the default code snippet and **paste the copied JSON**.
10. Click the **Refresh icon** next to the visual map to update the state machine diagram.

Understanding the State Machine Flow

- The state machine starts and waits for a **specific time period** (defined by the Timer state).
 - After the wait time, it triggers **an email reminder** using the **Lambda function**.
 - The state machine coordinates all steps in the serverless application workflow.
-

Stage 3C - Configure the State Machine

Now, we need to modify the state machine to use the **email Lambda function** we created earlier.

Steps to configure the state machine:

1. In the **ASL code (left panel of Step Functions Console)**, locate the **EmailOnly** definition.
 2. Find the placeholder **EMAIL_LAMBDA_ARN**.
 3. Replace **EMAIL_LAMBDA_ARN** with the **ARN of your email_reminder_lambda function** (which you saved in Stage 2).
 4. Scroll down and click **Next**.
 5. Enter **PetCuddleOTron** as the **State Machine Name**.
 6. Under **Permissions**, choose **"Use an existing role"** and select **StateMachineRole** from the dropdown.
 7. Under **Logging**, change **Log Level** to **All**.
 8. Scroll to the bottom and click **Create State Machine**.
 9. Once created, copy the **ARN of the state machine** (found in the top-left corner).
 10. Save this ARN as your **State Machine ARN** for later use.
-

Stage 3 - Completion

At this point, you have successfully:

- ✓ Created an **IAM Role** for the state machine with permissions for logging, Lambda, and SNS.
- ✓ Created and configured the **State Machine** that controls the workflow of the serverless application.

Next, in **Stage 4**, you will set up **API Gateway** and another **Lambda function** to serve as the front end for the application. 🚀

The screenshot displays the AWS Step Functions console. The top navigation bar shows 'Step Functions' and 'State machines'. The left sidebar contains 'Step Functions', 'State machines', 'Activities', and 'Developer resources'. The main content area shows a list of state machines with one entry: 'PetCuddleOTron', which is of type 'Standard' and has a status of 'Active'. Below this, the details for 'PetCuddleOTron' are shown, including its ARN, IAM role ARN, type, status, creation date, and X-Ray tracing status. The 'Executions' tab is selected, showing a table of executions with columns for Name, Status, Start Time, End Time, Duration, Version, and Alias. Three executions are listed, all with a status of 'Running'.

Name	Status	Start Time (local)	End Time (local)	Duration	Version	Alias
5a6e728b-f460-49bd-888c-a3489b5154fe	Running	Mar 1, 2025, 22:00:33	-	-	-	-
5354d454-b085-4ec5-94d9-115dd76675...	Running	Mar 1, 2025, 22:00:33	-	-	-	-
9b3b9f6d-b042-4de3-a0e1-13d4f844bd9f	Running	Mar 1, 2025, 22:00:32	-	-	-	-

● STAGE 4 : Implement the API Gateway, API and supporting lambda function


In this stage, you will set up the **API layer** for the serverless application. This API acts as the bridge between the **browser-based front end** (hosted on S3) and the **backend services** running on AWS.

The API is powered by **API Gateway**, which will handle incoming requests and pass them to a **Lambda function** (`api_lambda`) for processing. This Lambda function will then **trigger the Step Functions state machine**, which controls the flow of the application.

Stage 4A - Create the API Lambda Function

First, we need to create the Lambda function that will **handle API Gateway requests**.

Steps to create the Lambda function:


1. Open the **Lambda Console**:
 [AWS Lambda](#)
2. Click **Create Function**.
3. Set the following details:
 - **Function Name:** `api_lambda`
 - **Runtime:** `Python 3.9`
4. Expand **Change default execution role**.
5. Select **Use an existing role**.
6. Choose **LambdaRole** from the dropdown.
7. Click **Create Function**.

This function will handle requests from API Gateway and start the **Step Functions workflow** when needed.

Stage 4B - Configure the Lambda Function

Now, we will **add the code** to the `api_lambda` function.

Steps to configure the Lambda function:


1. Scroll down to the **Function code** section.
2. Delete all existing code from the `lambda_function` text box.
3. Open this link in a new tab:
 [API Lambda Function Code](#)
4. Copy all the code from the file.
5. Return to the **Lambda console** and **paste the copied code** into the function.
6. Locate the placeholder `YOUR_STATEMACHINE_ARN` in the code.
7. Replace `YOUR_STATEMACHINE_ARN` with the **State Machine ARN** that you noted down in **Stage 3**.
8. Click **Deploy** to save the function.

Now, this Lambda function will process API Gateway requests and start the **state machine execution**.

Stage 4C - Create the API Gateway

With the `api_lambda` function ready, we can now create the **API Gateway** that will serve as the **entry point** for our application.

Steps to create the API Gateway:

1. Open the **API Gateway Console**:
 [API Gateway](#)
 2. Click **APIs** on the left menu.
 3. Locate the **REST API** box and click **Build**.
 - Be careful **not to select the other API types (HTTP, WebSocket, etc.)**.
 4. If a **popup appears**, click **OK** to dismiss it.
 5. Under **Create new API**, ensure **New API** is selected.
 6. Set the following details:
 - **API Name:** `petcuddleotron`
 - **Endpoint Type:** `Regional`
 7. Click **Create API**.
-

Stage 4D - Create an API Resource

Now, we need to create an **API resource** that represents an endpoint within API Gateway.

Steps to create the resource:


1. Click the **Actions** dropdown and select **Create Resource**.
2. Set the following details:
 - **Resource Name:** `petcuddleton`
 - **Do NOT** check the **Configure as proxy resource** box.
 - **Check** the box for **Enable API Gateway CORS**.
3. Click **Create Resource**.

CORS (Cross-Origin Resource Sharing) is necessary because the front end (hosted on S3) will be calling this API. Without CORS, the browser will block the request.

Stage 4E - Create an API Method

Next, we will create a **POST method** for the `/petcuddleton` resource. This method will be responsible for **handling API requests** from the front end.

Steps to create the method:

1. Make sure `/petcuddleton` is selected in the left panel.
2. Click the **Actions** dropdown and select **Create Method**.
3. In the small dropdown below `/petcuddleton`, select **POST**, then click the  (tick mark).
4. Set the following details:
 - **Integration Type:** `Lambda Function`
 - **Lambda Region:** `us-east-1`
 - **Lambda Function:** Start typing `api_lambda`, then select it from the autocomplete list.
 - **Ensure** you select `api_lambda` and **not** `email_reminder_lambda`.
5. Make sure the following options are **enabled**:
 - ☒ **Use Default Timeout**
 - ☒ **Use Lambda Proxy Integration** (This ensures that all API data is forwarded correctly to Lambda).
6. Click **Save**.
7. You may see a **permission confirmation dialog**. Click **OK** to allow API Gateway to invoke the Lambda function.

Now, API Gateway has a **POST method** that connects to the `api_lambda` function.

Stage 4F - Deploy the API

Now that the API is configured, we need to **deploy** it so that it becomes accessible via an **Invoke URL**.

Steps to deploy the API:

1. Click the **Actions** dropdown and select **Deploy API**.
 2. Set the following details:
 - **Deployment Stage:** `New Stage`
 - **Stage Name:** `prod`
 - **Stage Description:** `prod`
 3. Click **Deploy**.
 4. At the top of the page, you will see an **Invoke URL**.
 - **Copy and save this URL**, as it will be needed in the next stage.
-

Stage 4 - Completion

At this point, you have successfully:

- ✓ Created the `api_lambda` function, which will start the state machine.
- ✓ Configured the **Lambda function** to handle API Gateway requests.
- ✓ Set up **API Gateway** with a **POST method** that routes requests to Lambda.
- ✓ **Deployed the API** to the `prod` stage.

Now, you have a fully functional **backend API** that the front end can communicate with.

In **Stage 5**, you will configure the **client-side application** (hosted on S3) to interact with the API Gateway. 🚀

Lambda > Functions > api_lambda

Successfully updated the function api_lambda.

api_lambda

ThrottleCopy ARNActions

Function overview

DiagramTemplate

api_lambda

Layers(0)

+ Add trigger

+ Add destination

Description

Last modified15 ago

Copied

ARNarn:aws:lambda:us-east-1:431797214645:function:api_lambda

Function URL

CodeTestMonitorConfigurationAliasesVersions

Code source

Upload from

lambda_function.py

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

282

283

284

285

286

287

288

289

290

291

292

293

294

295

296

297

298

299

300

301

302

303

304

305

306

307

308

309

310

311

312

313

314

315

316

317

318

319

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

339

340

341

342

343

344

345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360

361

362

363

364

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

381

382

383

384

385

386

387

388

389

390

391

392

393

394

395

396

397

398

399

400

401

402

403

404

405

406

407

408

409

410

411

412

413

414

415

416

417

418

419

420

421

422

423

424

425

426

427

428

429

430

431

432

433

434

435

436

437

438

439

440

441

442

443

444

445

446

447

448

449

450

451

452

453

454

455

456

457

458

459

460

461

462

463

464

465

466

467

468

469

470

471

472

473

474

475

476

477

478

479

480

481

482

483

484

485

486

487

488

489

490

491

492

493

494

495

496

497

498

499

500

501

502

503

504

505

506

507

508

509

510

511

512

513

514

515

516

517

518

519

520

521

522

523

524

525

526

527

528

529

530

531

532

533

534

535

536

537

538

539

540

541

542

543

544

545

546

547

548

549

550

551

552

553

554

555

556

557

558

559

560

561

562

563

564

565

566

567

568

569

570

571

572

573

574

575

576

577

578

579

580

581

582

583

584

585

586

587

588

589

590

591

592

593

594

595

596

597

598

599

600

601

602

603

604

605

606

607

608

609

610

611

612

613

614

615

616

617

618

619

620

621

622

623

624

625

626

627

628

629

630

631

632

633

634

635

636

637

638

639

640

641

642

643

644

645

646

647

648

649

650

651

652

653

654

655

656

657

658

659

660

661

662

663

664

665

666

667

668

669

670

671

672

673

674

675

676

677

678

679

680

681

682

683

684

685

686

687

688

689

690

691

692

693

694

695

696

697

698

699

700

701

702

703

704

705

706

707

708

709

710

711

712

713

714

715

716

717

718

719

720

721

722

723

724

725

726

727

728

729

730

731

732

733

734

735

736

737

738

739

740

741

742

743

744

745

746

747

748

749

750

751

752

753

754

755

756

757

758

759

760

761

762

763

764

765

766

767

768

769

770

771

772

773

774

775

776

777

778

779

780

781

782

783

784

785

786

787

788

789

790

791

792

793

794

795

796

797

798

799

800

801

802

803

804

805

806

807

808

809

810

811

812

813

814

815

816

817

818

819

820

821

822

823

824

825

826

827

828

829

830

831

832

833

834

835

836

837

838

839

840

841

842

843

844

845

846

847

848

849

850

851

852

853

854

855

856

857

858

859

860

861

862

863

864

865

866

867

868

869

870

871

872

873

874

875

876

877

878

879

880

881

882

883

884

885

886

887

888

889

890

891

892

893

894

895

896

897

898

899

900

901

902

903

904

905

906

907

908

909

910

911

912

913

914

915

916

917

918

919

920

921

922

923

924

925

926

927

928

929

930

931

932

933

934

935

936

937

938

939

940

941

942

943

944

945

946

947

948

949

950

951

952

953

954

955

956

957

958

959

960

961

962

963

964

965

966

967

968

969

970

971

972

973

974

975

976

977

978

979

980

981

982

983

984

985

986

987

988

989

990

991

992

993

994

995

996

997

998

999

1000

1001

1002

1003

1004

1005

1006

1007

1008

1009

1010

1011

1012

1013

1014

1015

1016

1017

1018

1019

1020

1021

1022

1023

1024

1025

1026

1027

1028

1029

1030

1031

1032

1033

1034

1035

1036

1037

1038

1039

1040

1041

1042

1043

1044

1045

1046

1047

1048

1049

1050

1051

1052

1053

1054

1055

1056

1057

1058

1059

1060

1061

1062

1063

1064

1065

1066

1067

1068

1069

1070

1071

1072

1073

1074

1075

1076

1077

1078

1079

1080

1081

1082

1083

1084

1085

1086

1087

1088

1089

1090

1091

1092

1093

1094

1095

1096

1097

1098

1099

1100

1101

1102

1103

1104

1105

1106

1107

1108

1109

1110

1111

1112

1113

1114

1115

1116

1117

1118

1119

1120

1121

1122

1123

1124

1125

1126

1127

1128

1129

1130

1131

1132

1133

1134

1135

1136

1137

1138

1139

1140

1141

1142

1143

1144

1145

1146

1147

1148

1149

1150

1151

1152

1153

1154

1155

1156

1157

1158

1159

1160

1161

1162

1163

1164

1165

1166

1167

1168

1169

1170

1171

1172

1173

1174

1175

1176

1177

1178

1179

1180

1181

1182

1183

1184

1185

1186

1187

1188

1189

1190

1191

1192

1193

1194

1195

1196

1197

1198

1199

1200

1201

1202

1203

1204

1205

1206

1207

1208

1209

1210

1211

1212

1213

1214

1215

1216

1217

1218

1219

1220

1221

1222

1223

1224

1225

1226

1227

1228

1229

1230

1231

1232

1233

1234

1235

1236

1237

1238

1239

1240

1241

1242

1243

1244

1245

1246

1247

1248

1249

1250

1251

1252

1253

1254

1255

1256

1257

1258

1259

1260

1261

1262

1263

1264

1265

1266

1267

1268

1269

1270

1271

1272

1273

1274

1275

1276

1277

1278

1279

1280

1281

1282

1283

1284

1285

1286

1287

1288

1289

1290

1291

1292

1293

1294

1295

1296

1297

1298

1299

1300

1301

1302

1303

1304

1305

1306

1307

1308

1309

1310

1311

1312

1313

1314

1315

1316

1317

1318

1319

1320

1321

1322

1323

1324

1325

1326

1327

1328

1329

1330

1331

1332

1333

1334

1335

1336

1337

1338

1339

1340

1341

1342

1343

1344

1345

1346

1347

1348

1349

1350

1351

1352

1353

1354

1355

1356

1357

1358

1359

1360

1361

1362

1363

1364

1365

1366

1367

1368

1369

1370

1371

1372

1373

1374

1375

1376

1377

1378

1379

1380

1381

1382

1383

1384

1385

1386

1387

1388

1389

1390

1391

1392

1393

1394

1395

1396

1397

1398

1399

1400

1401

1402

1403

1404

1405

1406

1407

1408

1409

1410

1411

1412

1413

1414

1415

1416

1417

1418

1419

1420

1421

1422

1423

1424

1425

1426

1427

1428

1429

1430

1431

1432

1433

1434

1435

1436

1437

1438

1439

1440

1441

1442

1443

1444

1445

1446

1447

1448

1449

1450

1451

1452

1453

1454

1455

1456

1457

1458

1459

1460

1461

1462

1463

1464

1465

1466

1467

1468

1469

1470

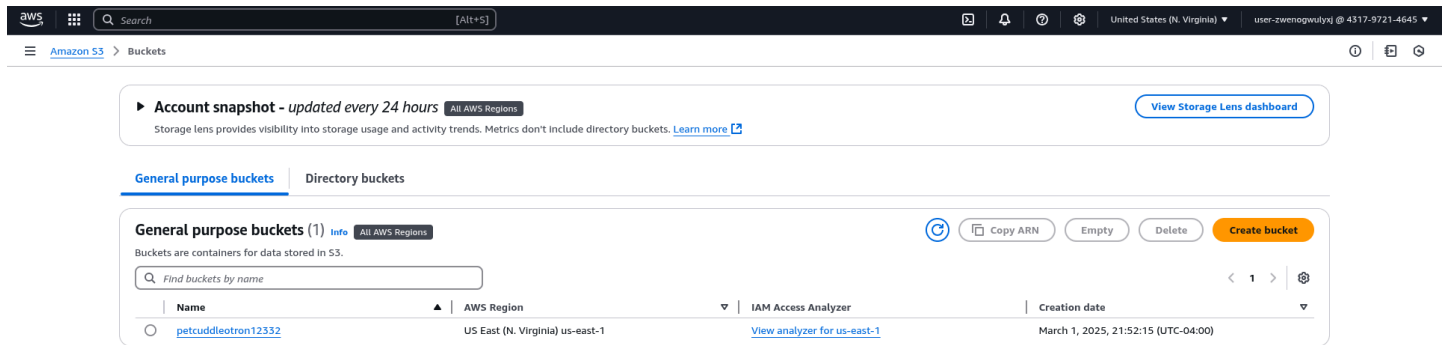
1471

1472

1473

1474

1475



● STAGE 5 : Implement the static frontend application and test functionality

In this stage, you will set up the **front-end** for the serverless application. The front end is a simple **static website** hosted on **Amazon S3**, which communicates with the API Gateway using JavaScript.

Stage 5A - Create an S3 Bucket

To host the front-end files, you need to create an **S3 bucket**.

Steps to create the bucket:

1. Open the **S3 Console**:
👉 [AWS S3](#)
2. Click **Create Bucket**.
3. Choose a **unique bucket name** (bucket names must be globally unique).
4. Make sure the **Region** is set to **US East (N. Virginia) (us-east-1)**.
5. Scroll down and **UNTICK** the box for **Block all public access**.
6. Tick the checkbox confirming that you understand the bucket could become public.
7. Scroll down and click **Create Bucket**.

This bucket will store the static files for the front-end application.

Stage 5B - Make the Bucket Public

Since this is a public website, you need to configure the bucket so that **anyone can access** the front-end files.

Steps to make the bucket public:

1. Open the **S3 Console** and go to the bucket you just created.
2. Click on the **Permissions** tab.
3. Scroll down to the **Bucket Policy** section and click **Edit**.

Copy and paste the following policy into the box:

json

CopyEdit

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicRead",
      "Effect": "Allow",
      "Principal": "*",
      "Action": ["s3:GetObject"],
      "Resource": ["REPLACEME_PET_CUDDLE_O_TRON_BUCKET_ARN/*"]
    }
  ]
}
```

- 4.
5. Replace **REPLACEME_PET_CUDDLE_O_TRON_BUCKET_ARN** with your **actual bucket ARN** (you can find it in the bucket details).
6. Click **Save Changes**.

Now, anyone can access the files in this bucket.

Stage 5C - Enable Static Website Hosting

To turn this S3 bucket into a **website**, you need to enable **static hosting**.

Steps to enable website hosting:

1. Go to the **Properties** tab of your bucket.
2. Scroll down to **Static website hosting**.
3. Click **Edit**.

4. Select **Enable**.
5. Choose **Host a static website**.

For both **Index Document** and **Error Document**, enter:

diff

CopyEdit

`index.html`

- 6.
7. Click **Save Changes**.
8. Scroll down again and **copy** the **Bucket Website Endpoint URL**.

This URL is where your front-end website will be accessible.

Stage 5D - Download and Edit the Front-End Files

Now, you need to download and modify the **front-end files** so that they can connect to your API Gateway.

Steps to download and edit the front-end files:

1. Download the front-end files from this link:
👉 [Front-End ZIP](#)
2. Extract the ZIP file. Inside, you will see these files:
 - `index.html` – The main web page.
 - `main.css` – Stylesheet for the page.
 - `whiskers.png` – An image used in the app.
 - `serverless.js` – JavaScript file that handles API requests.
3. Open `serverless.js` in a **code editor** (e.g., VS Code, Notepad++, or any text editor).
4. Find the placeholder `REPLACEME_API_GATEWAY_INVOKE_URL`.
5. Replace it with your **API Gateway Invoke URL** (from Stage 4F).
6. At the end of the URL, add `/petcuddleotron`.

Example:

bash

CopyEdit

`https://your-api-id.execute-api.us-east-1.amazonaws.com/prod/petcuddleotron`

○

7. Save the file.

This ensures that when users click buttons on the front end, the requests are sent to the correct API Gateway endpoint.


Stage 5E - Upload and Test the Front-End

Now, you will upload the **edited files** to your S3 bucket and test the application.

Steps to upload the files:

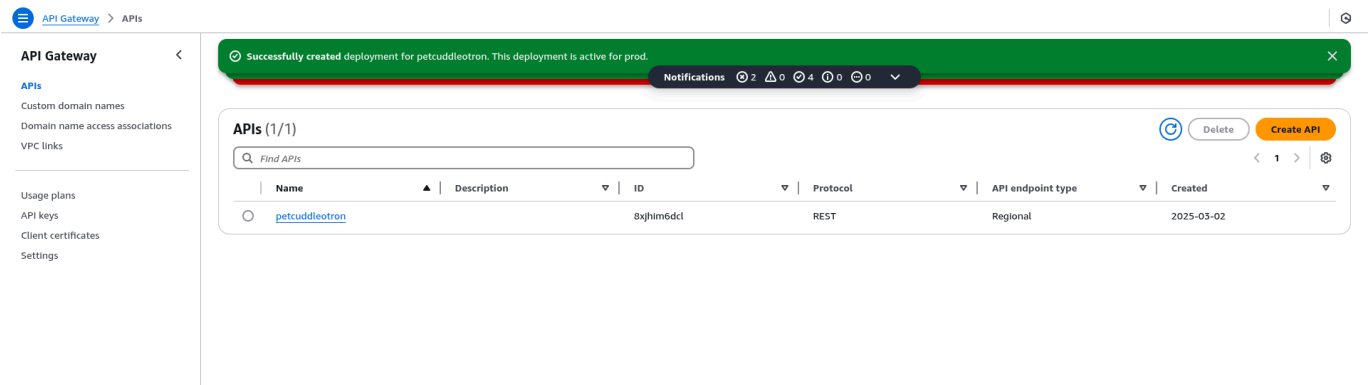
1. Go to your **S3 bucket**.
2. Click on the **Objects** tab.
3. Click **Upload**.
4. Drag and drop all **four files** (`index.html`, `main.css`, `whiskers.png`, and the modified `serverless.js`) into the upload area.
5. Click **Upload** and wait for it to complete.
6. Click **Exit** and verify that all four files are in the bucket.

Steps to test the application:

1. Open the **Bucket Website Endpoint URL** (copied in Stage 5C) in a web browser.
2. You should see a **simple web page** with buttons and input fields.
3. Enter a **time delay** (e.g., 120 seconds).
4. Enter a **message** (e.g., "HUMAN, COME HOME NOW!").
5. Enter the **email address** that you verified earlier in **SES**.
6. Before clicking the button, open the **Step Functions Console** in a new tab:
 [Step Functions Console](#)
7. Click on **PetCuddleOTron**.
8. Click on the **Executions** tab (you should see no executions yet).
9. Go back to the **web application** and click the "**Email Minion**" button.
10. Switch back to the **Step Functions Console** and click **Refresh**.
11. Click on the new execution and watch the flow:
 - **Timer state** will activate (waits for the delay).
 - **Email state** will trigger the **email Lambda function**.
 - The state machine will **end** after processing.

Check the logs:

1. Click on **Execution Input** in Step Functions to see the data sent from the web app.
 2. Click on **Logging** to review CloudWatch logs.
-



Stage 5 - Completion

At this point, you have a **fully functional serverless application**:

- ✓ **Front-end HTML & JavaScript** is hosted on **S3**
- ✓ **JavaScript communicates with API Gateway**
- ✓ **API Gateway routes requests to `api_lambda`**
- ✓ **`api_lambda` triggers Step Functions**
- ✓ **Step Functions sends an email using Lambda & SES**
- ✓ **The application runs entirely serverless** 🚀


This completes the main setup! 🎉

In **Stage 6**, you will clean up all AWS resources used in this demo.

file:///home/alejandroM/Documents/DevOps-Project2025/AWS-MINIPROJECT/Screenshot%20from%202025-03-01%2022-15-45.png

petcuddletron12332.s3-website-us-east-1.amazonaws.com

Documentation Blog Bug tracker GitHub organization



Pet Cuddle-O-Tron

^^ Whiskers Commands You!! ^^

Submitted. But check the result below!

These are always needed:

120

asasa

Email address is required to send the human an email

39@gmail.com

Human Minion Hailing Control

Email Minion

{ "Status": "Success" }

←

📁

⚠

🗑

|

✉

📁

⋮

Whiskers Commands You to attend!

Inbox x

asasa

39@gmail.com

3

asasasdfsds

39@gmail.com

asasasdfsds

asasasdfsds

39@gmail.com via amazonses.com

to me ▼

asasasdfsds

