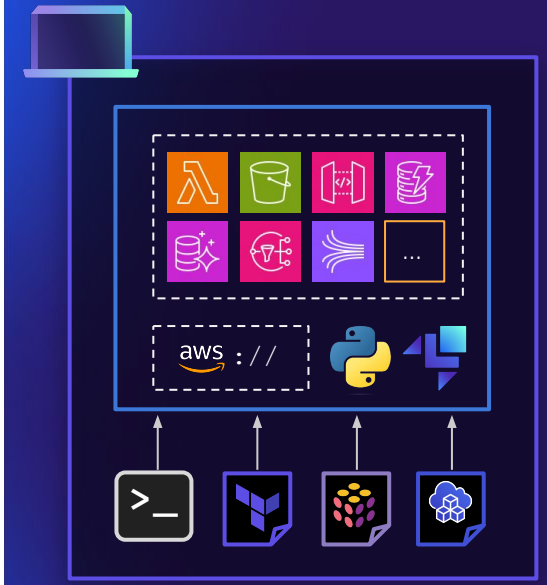


How LocalStack is recreating AWS with Python

Dr. Thomas Rausch
Co-Founder at LocalStack

@thrauat

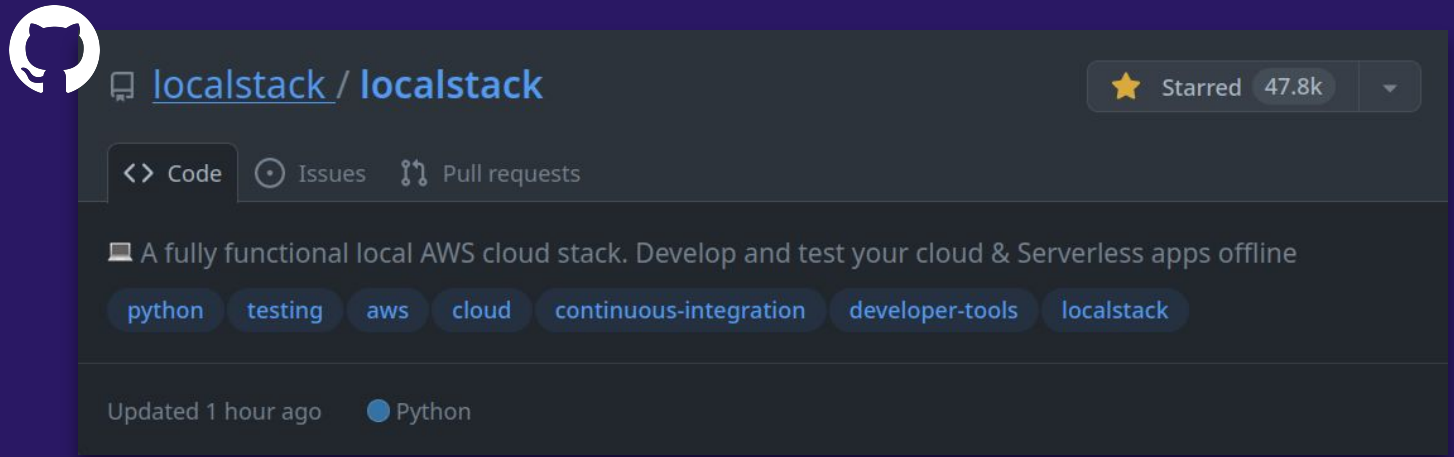
LocalStack

EuroPython 2023
Prague & Remote

LocalStack is

a platform for cloud application development and testing

LocalStack is an AWS emulator



LocalStack is

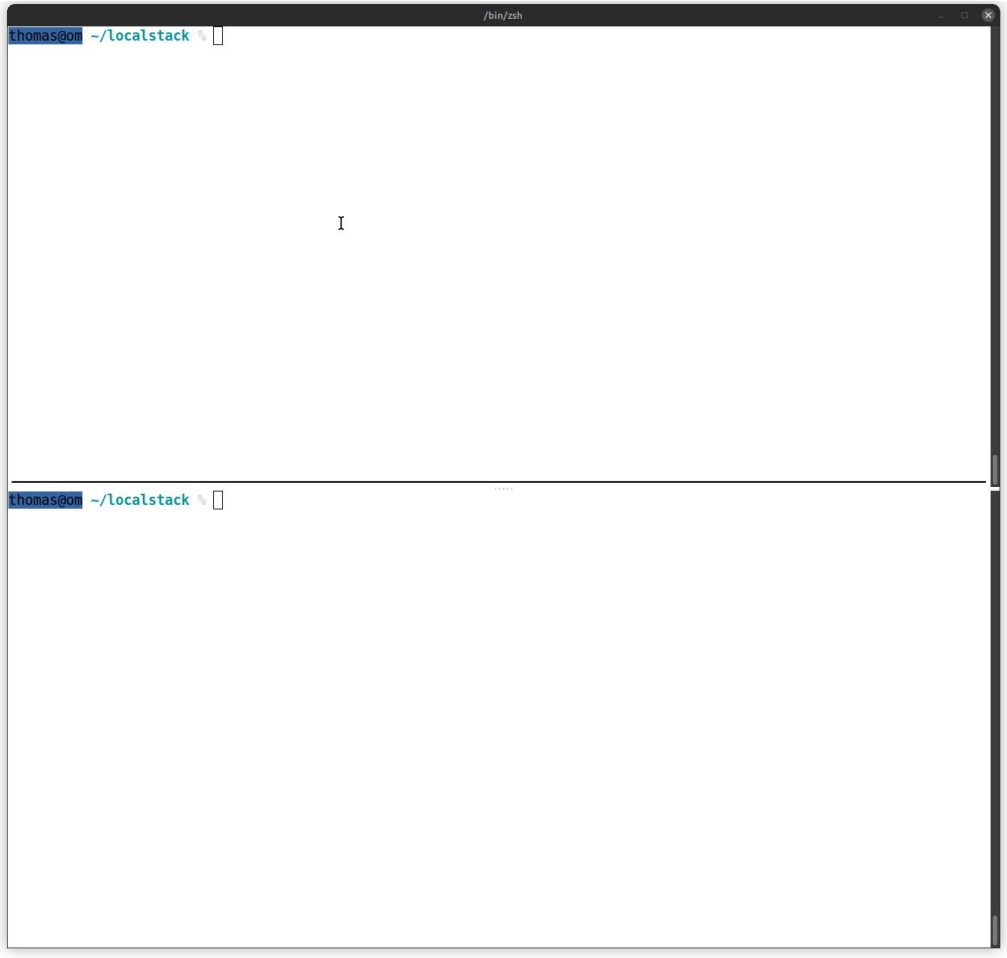
a drop-in replacement for AWS
that runs on your local machine

HashiCorp Terraform | aws

LocalStack

main.tf

```
provider "aws" {
  region = "us-east-1"
}

resource "aws_s3_bucket" "bucket" {
  bucket = "your-bucket-name"
}

resource "aws_s3_bucket_notification" "bucket_notification" {
  bucket = aws_s3_bucket.bucket.id

  queue {
    queue_arn = aws_sqs_queue.queue.arn
    events    = ["s3:ObjectCreated:*"]
  }
}

resource "aws_sqs_queue" "queue" {
  name = "s3-event-notification-queue"

  policy = <<POLICY
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": "sqs:SendMessage",
```



docs.localstack.cloud/applications

Serverless Container-based APIs with Amazon ECS & API Gateway

Pro

serverless-containers · security · identity · compliance

Deploy a Full-Stack Serverless Web application, and deploy it with Terraform & CloudFormation on LocalStack

Full-Stack application with AWS Lambda, DynamoDB & S3 for shipment validation

spring-boot · lambda-trigger

Configure a CRUD web application for shipment validation & listing, and deploy it with Terraform on LocalStack

Step-up Authentication using Amazon Cognito

Pro

step-up-auth · rule-based-authentication · localsurf

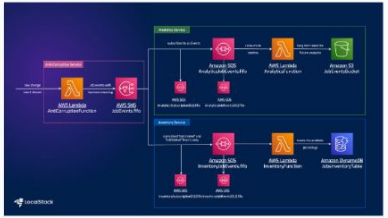
Setup a Step-up Authentication workflow for a higher level of security, deployed using Cloud Development Kit on LocalStack

Serverless microservices with Amazon API Gateway, DynamoDB, SQS, and Lambda

Pro

serverless · microservices

Serverless microservices for managing friend state in gaming services asynchronously deployed using Cloud Development Kit on LocalStack



Event-driven architecture with Amazon SNS FIFO, DynamoDB, Lambda, and S3

Pro

serverless · event-driven-architecture

Event-driven architecture demonstrating a recruiting agency application deployed using Serverless Application Model on LocalStack

Note-Taking application using AWS SDK for JavaScript

Pro

serverless

Serverless Note-Taking application demonstrating a Web Application built using Lambda, API Gateway, and DynamoDB deployed using CDK on LocalStack

Filters

Services

- API Gateway
- AppSync
- Athena
- CloudFront
- Cognito
- CloudFormation
- CloudWatch Logs
- Dynamo DB
- EventBridge
- ECR
- ECS
- fargate
- Glue
- IAM
- Kinesis
- Lambda
- RDS
- S3
- Secrets Manager
- SES
- SageMaker
- SNS
- SQS
- Systems Manager
- Step Functions
- Transcribe

Infrastructure Provisioned

- AWS CLI
- CDK
- Cloudformation
- SAM
- Serverless
- Terraform

Programming language

How do we do it?

How do we do it?

- Use Python!

Search the docs

Examples using the Docker Engine SDKs and Docker API

After you install Docker, you can install the Go or Python SDK and also try out the Docker Engine API.

Each of these examples show how to perform a given Docker operation using the Go and Python SDKs and the HTTP API using `curl` .

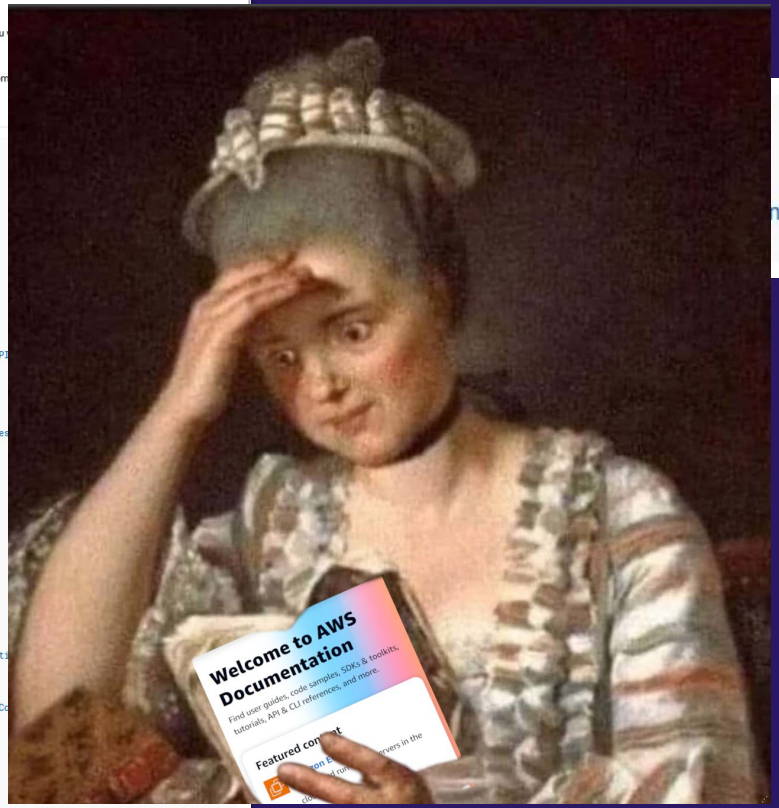
Run a container

This first example shows how to run a container using the Docker API. On the command line, you
just as easy to do from your own apps too.

This is the equivalent of typing `docker run alpine echo hello world` at the command prom

Go | Python | HTTP

```
package main

import (
        "context"
        "io"
        "os"

        "github.com/docker/docker/api/types"
        "github.com/docker/docker/api/types/container"
        "github.com/docker/docker/client"
        "github.com/docker/docker/pkg/stdcopy"
)

func main() {
        ctx := context.Background()
        cli, err := client.NewClientWithOpts(client.FromEnv, client.WithAPI
        if err != nil {
                panic(err)
        }
        defer cli.Close()

        reader, err := cli.ImagePull(ctx, "docker.io/library/alpine", types
        if err != nil {
                panic(err)
        }

        defer reader.Close()
        io.Copy(os.Stdout, reader)

        resp, err := cli.ContainerCreate(ctx, &container.Config{
                Image: "alpine",
                Cmd:   []string{"echo", "hello world"},
                Tty:   false,
        }, nil, nil, nil, "")
        if err != nil {
                panic(err)
        }

        if err := cli.ContainerStart(ctx, resp.ID, types.ContainerStartOpti
                panic(err)
        }

        statusCh, errCh := cli.ContainerWait(ctx, resp.ID, container.WaitCo
        select {
        case err := <-errCh:
                if err != nil {
                        panic(err)
                }
        case <-statusCh:
        }

        out, err := cli.ContainerLogs(ctx, resp.ID, types.ContainerLogsOptions{ShowStdout: true})
        if err != nil {
                panic(err)
        }

        stdcopy.StdCopy(os.Stdout, os.Stderr, out)
}
```




```
n("alpine", ["echo", "hello", "world"]))
```

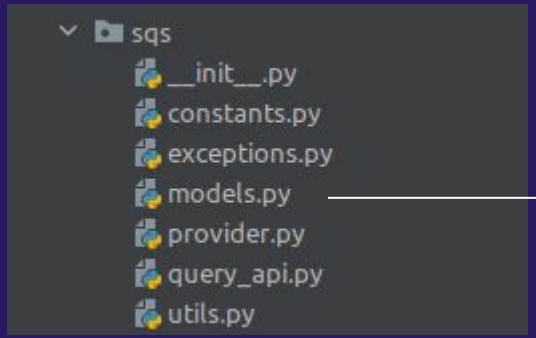
https://docs.docker.com/engine/api/sdk/examples/

How do we do it?

- Use Python!
- Reduce complexity

We can reduce complexity

- Everything runs locally!
- Fewer distributed system challenges
- Example: full Python SQS implementation in ~2k LoC




```python
class SqsQueue:
    name: str
    region: str
    account_id: str

    attributes: QueueAttributeMap
    tags: TagMap

    purge_in_progress: bool
    purge_timestamp: Optional[ float]

    delayed: Set[SqsMessage]
    inflight: Set[SqsMessage]
    receipts: Dict[ str, SqsMessage]
```

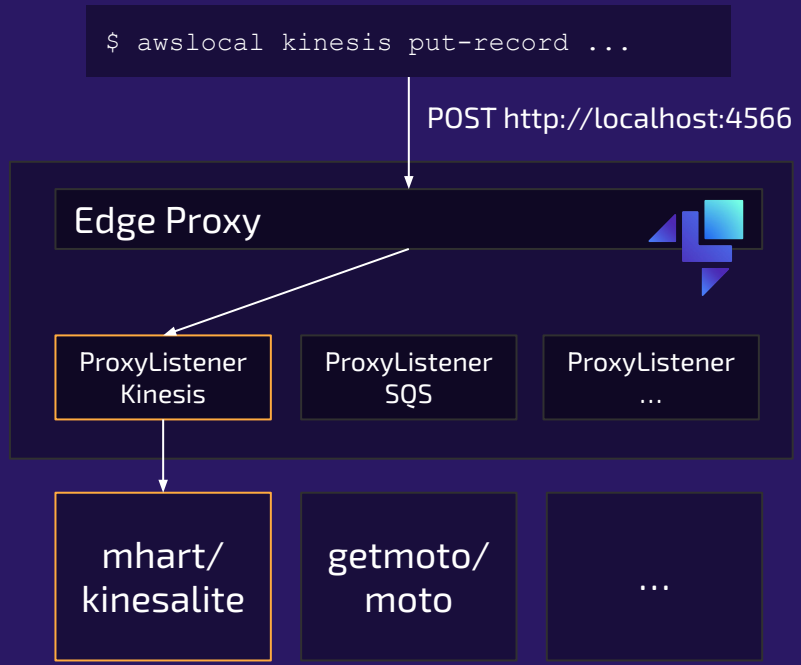
How do we do it?

- Use Python!
- Reduce complexity
- Leverage the open source AWS ecosystem
- Standardize the repeatable parts (service emulators)
  - Development
  - Testing
  - Maintenance
- Embrace pluggability

Service emulators

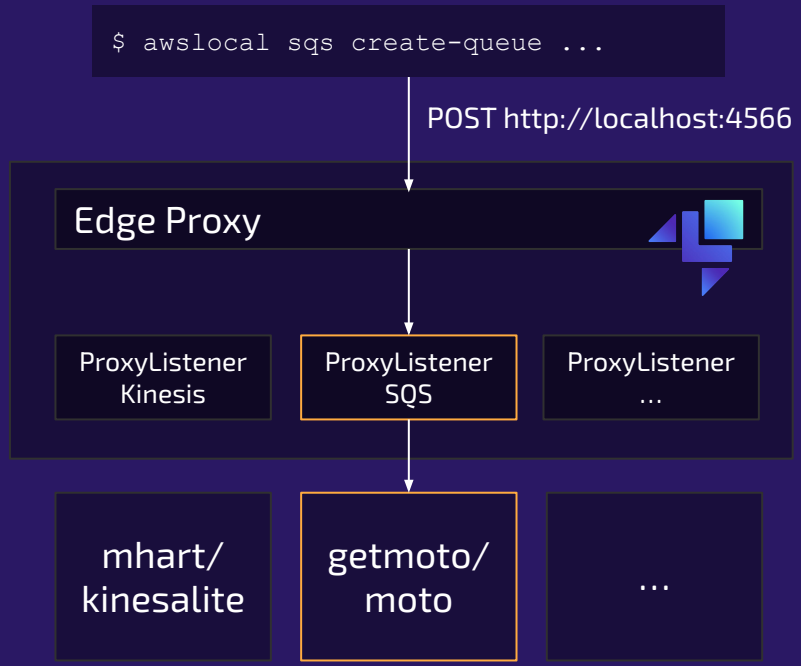
Service emulators (how it used to be)

```
class ProxyListenerKinesis (ProxyListener):

    def forward_request (self, method, path, data, headers):
        data, encoding_type =  self.decode_content(data  or "{}", True)
        action = headers.get( "X-Amz-Target" , "").split(".")[-1]

        if action == "SubscribeToShard" :
            result = subscribe_to_shard(data, headers)
            return result

        if random.random() < config.KINESIS_ERROR_PROBABILITY:
            if action in ["PutRecord", "PutRecords"]:
                return kinesis_error_response(data, actio encoding_type)

        if config.KINESIS_PROVIDER ==  "kinesalite" :
            return self.forward_request_kinesalite(
                method, path, data, headers, action, encoding_ty
            )

        return True
```

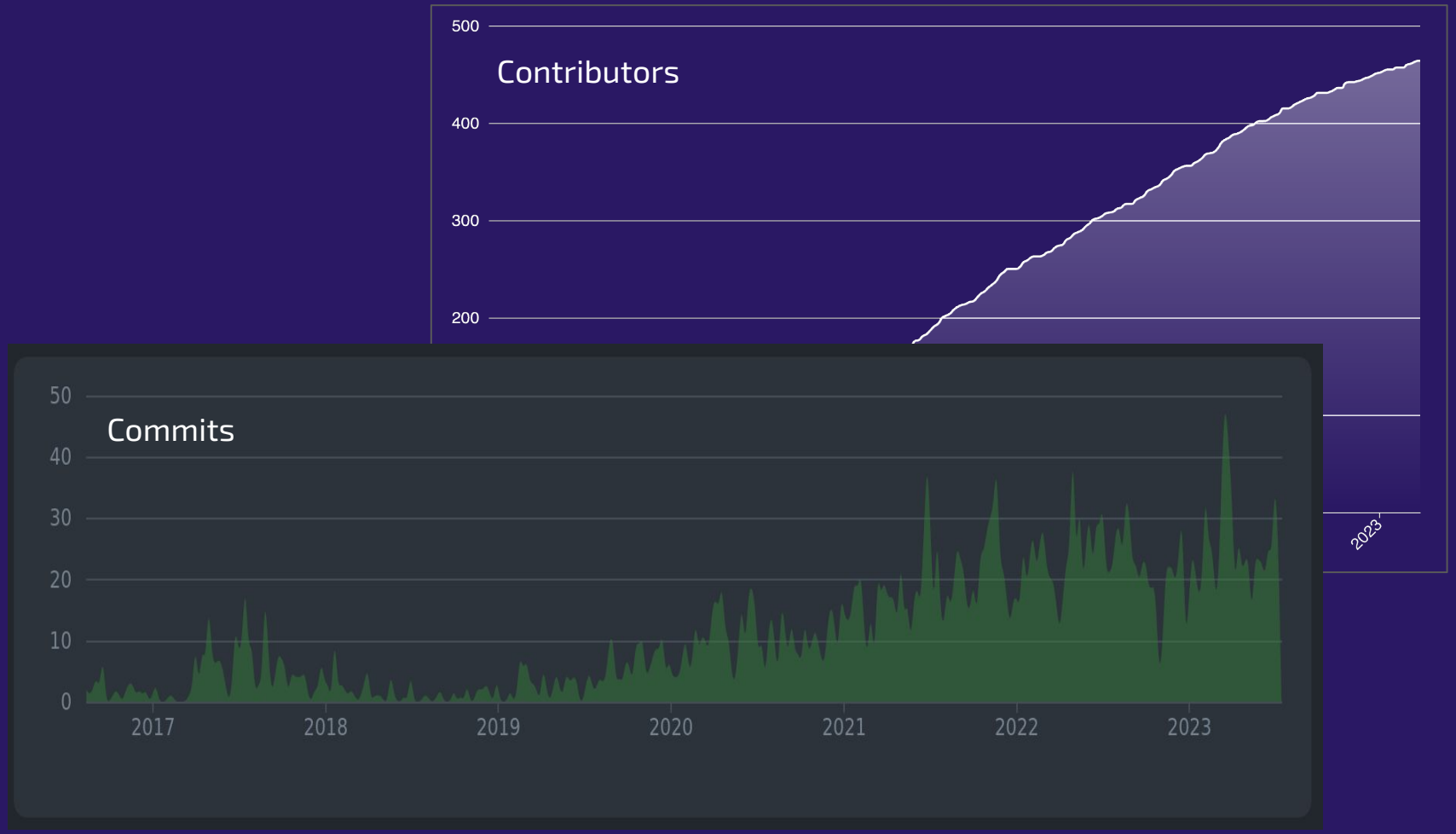


Adding dev tool functionality

```
class ProxyListenerSQS(PersistingProxyListener):

    def forward_request(self, method, path, data, headers):
        if method == "OPTIONS":
            return 200

        req_data = parse_request_data(method, path, data)

        if req_data:
            action = req_data.get("Action")

            if action == "CreateQueue":
                req_data = self.fix_missing_tag_values(req_data)
                changed_attrs = _fix_dlq_arn_in_attributes(req_data)
                if changed_attrs:
                    return _get_attributes_forward_request(
                        method, path, headers, req_data,
                    )

            elif action == "DeleteQueue":
                queue_url = _queue_url(path, req_data, headers)
                QUEUE_ATTRIBUTES.pop(queue_url,None)
                sns_listener.unsubscribe_sqs_queue(queue_url)

            elif action == "SetQueueAttributes":
                queue_url = _queue_url(path, req_data, headers)
                if SQS_BACKEND_IMPL == "elasticmq":
```

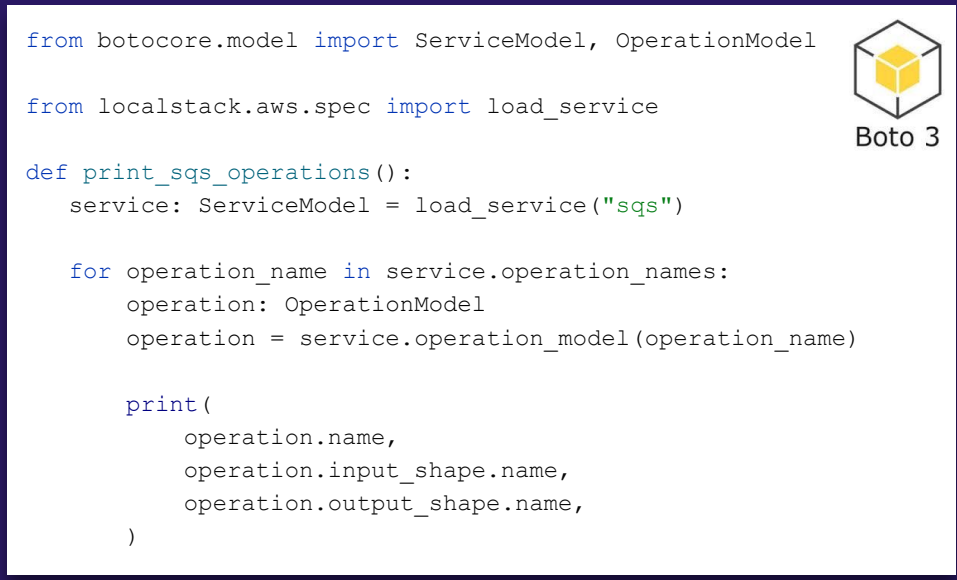


Growing number of service integrations


Contributors
500
400
300
200
2023
Commits
50
40
30
20
10
0
2017
2018
2019
2020
2021
2022
2023

Service emulators (how it’s now)

Leveraging AWS open source tools

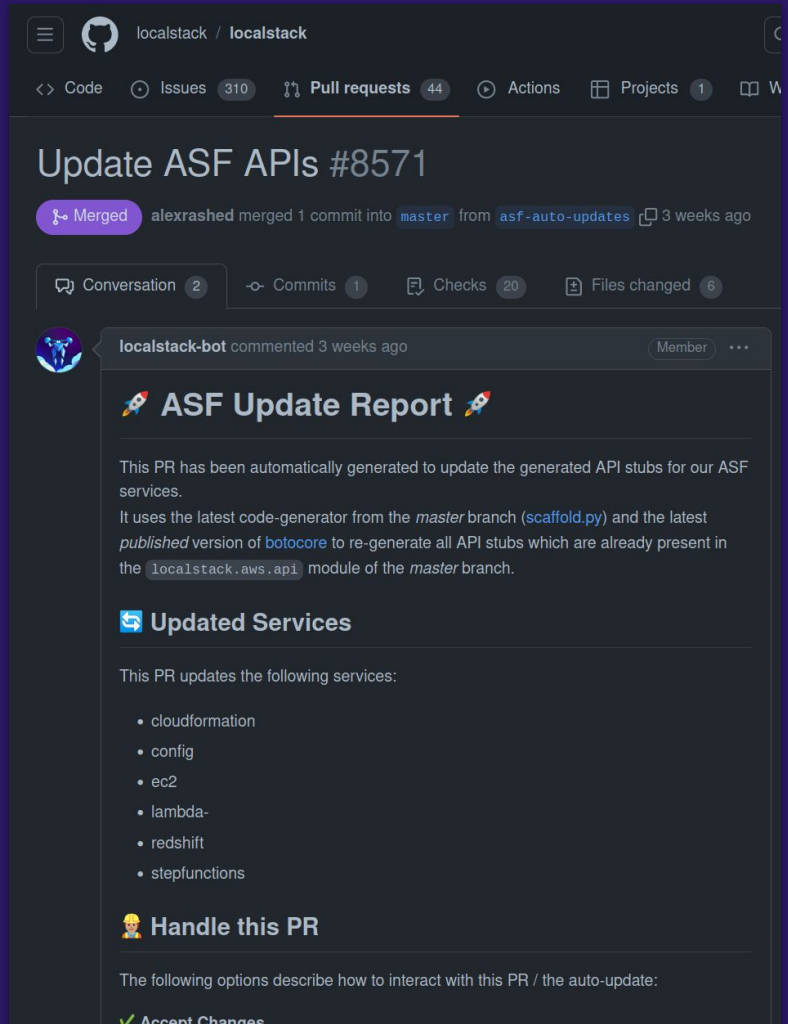


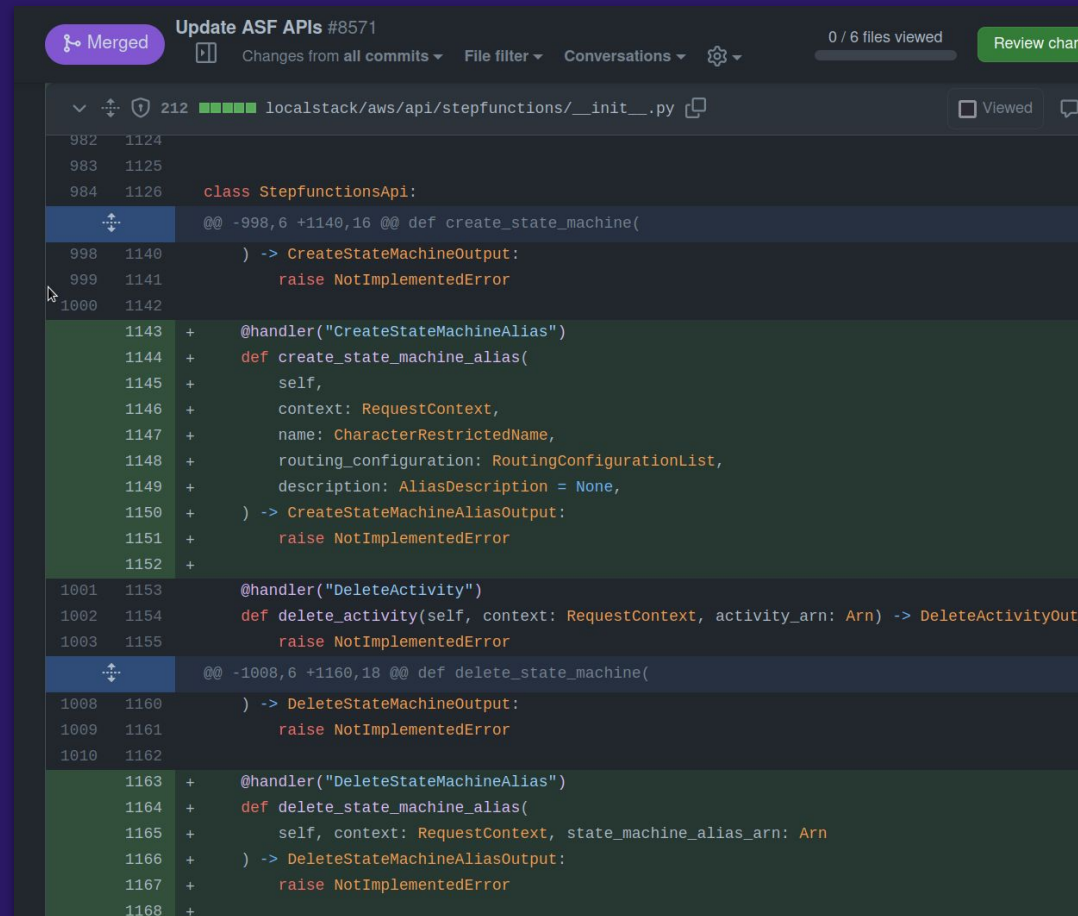
https://github.com/boto/botocore/blob/develop/botocore/data/sqs/2012-11-05/service-2.json

```python
def generate_code(service_name: str) -> str:
    model: ServiceModel = load_service(service_name)

    output = io.StringIO()

    generate_service_types(output, model)
    generate_service_api(output, model)

    code = output.getvalue()

    import autoflake
    import isort
    import black

    # try to remove unused imports
    code = autoflake.fix_code(code, remove_all_unused_imports=True)

    # try to format with black
    code = black.format_str(code, mode=black.FileMode(line_length=100))

    # try to sort imports
    code = isort.code(code, config=isort.Config(profile="black",
line_length=100))
```

/bin/zsh

```
thomas@ninox ~/workspace/localstack/localstack
 %
```


localstack / localstack
Code
Issues 310
Pull requests 44
Actions
Projects 1
Update ASF APIs #8571
Merged
alexrashed merged 1 commit into master from asf-auto-updates 3 weeks ago
Conversation 2
Commits 1
Checks 20
Files changed 6
localstack-bot commented 3 weeks ago
Member
🚀 ASF Update Report 🚀
This PR has been automatically generated to update the generated API stubs for our ASF services.
It uses the latest code-generator from the master branch (scaffold.py) and the latest published version of botocore to re-generate all API stubs which are already present in the localstack.aws.api module of the master branch.
🔄 Updated Services
This PR updates the following services:
cloudformation
config
ec2
lambda-
redshift
stepfunctions
👷 Handle this PR
The following options describe how to interact with this PR / the auto-update:


Merged
Update ASF APIs #8571
Changes from all commits
File filter
Conversations
0 / 6 files viewed
Review chan
212 localstack/aws/api/stepfunctions/__init__.py
Viewed
class StepfunctionsApi:
@@ -998,6 +1140,16 @@ def create_state_machine(
) -> CreateStateMachineOutput:
raise NotImplementedError
@handler("CreateStateMachineAlias")
def create_state_machine_alias(
self,
context: RequestContext,
name: CharacterRestrictedName,
routing_configuration: RoutingConfigurationList,
description: AliasDescription = None,
) -> CreateStateMachineAliasOutput:
raise NotImplementedError
@handler("DeleteActivity")
def delete_activity(self, context: RequestContext, activity_arn: Arn) -> DeleteActivityOut
raise NotImplementedError
@@ -1008,6 +1160,18 @@ def delete_state_machine(
) -> DeleteStateMachineOutput:
raise NotImplementedError
@handler("DeleteStateMachineAlias")
def delete_state_machine_alias(
self, context: RequestContext, state_machine_alias_arn: Arn
) -> DeleteStateMachineAliasOutput:
raise NotImplementedError

350 services

13,708 API operations

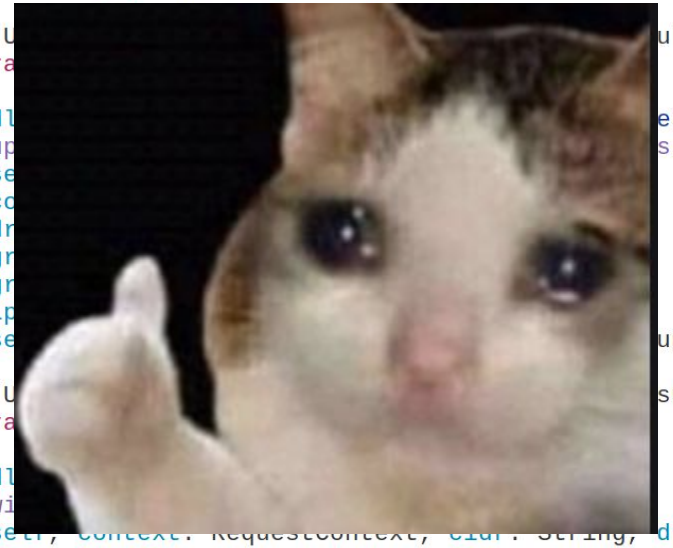
80 MB JSON specs

LocalStack:

93 services (and growing)

/bin/zsh

```
23878         ) -> UnmonitorInstancesResult:
23879             raise NotImplementedError
23880
23881         @handler("UpdateSecurityGroupRuleDescriptionsEgress")
23882         def update_security_group_rule_descriptions_egress(
23883             self,
23884             context: RequestContext,
23885             dry_run: Boolean = None,
23886             group_id: SecurityGroupId = None,
23887             group_name: SecurityGroupName = None,
23888             ip_permissions: IpPermissionList = None,
23889             security_group_rule_descriptions: SecurityGroupRuleDescriptionList
      = None,
23890         ) -> U                                                    ult:
23891             ra
23892
23893         @handl                                                    ess")
23894         def up                                                    ss(
23895             se
23896             co
23897             dr
23898             gr
23899             gr
23900             ip
23901             se                                                    upRuleDescriptionList
      = None,
23902         ) -> U                                                    sult:
23903             ra
23904
23905         @handl
23906         def wi
23907             self, context: RequestContext, cidr: String, dry_run: Boolean = Non
      e
23908         ) -> WithdrawByoipCidrResult:
23909             raise NotImplementedError
```



23909 lines

(END)

API Stub

```
class SqsApi:

    service = "sqs"
    version = "2012-11-05"

    @handler("CreateQueue")
    def create_queue(
        self,
        context: RequestContext,
        queue_name: String,
        attributes: QueueAttributeMap = None,
        tags: TagMap = None,
    ) -> CreateQueueResult:
        raise NotImplementedError
```

implements

Implementation

```
class SqsProvider(SqsApi):

    @handler("CreateQueue")
    def create_queue(
        self,
        context: RequestContext,
        queue_name: String,
        attributes: QueueAttributeMap = None,
        tags: TagMap = None,
    ) -> CreateQueueResult:
        fifo =
attributes.get(QueueAttributeName.FifoQueue)
        queue = SqsQueue(...)
        # ...
```

Examples:

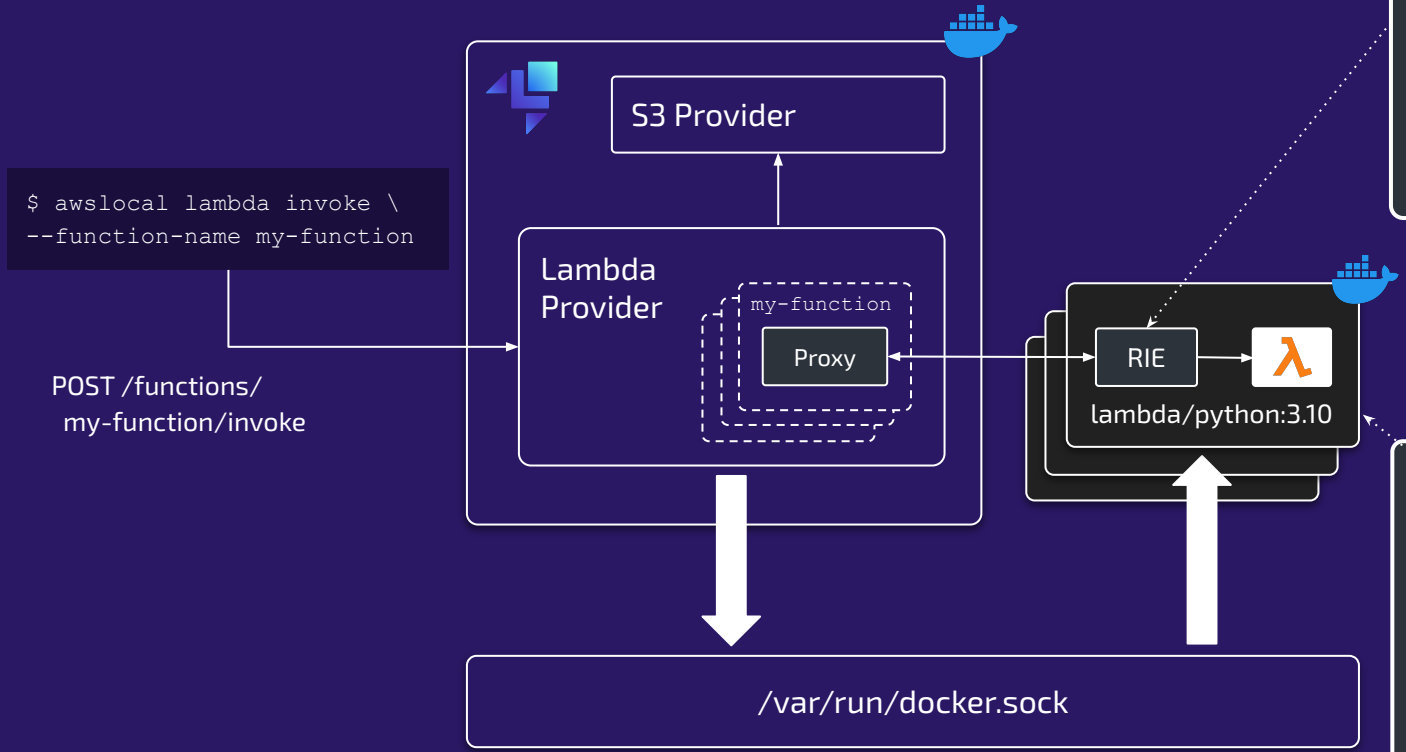
```
class ActiveMQPackage(Package):
    def __init__(self):
        super().__init__("ActiveMQ", "5.16.6")

    def get_versions(self) -> List[str]:
        return ["5.16.6"]

    def _get_installer(self, version: str) -> PackageInstaller:
        return ActiveMQPackageInstaller("active-mq", version)
```

uses or provides

Backend Service

Open AWS container tools

Example: Lambda



AWS Lambda Runtime Interface Emulator

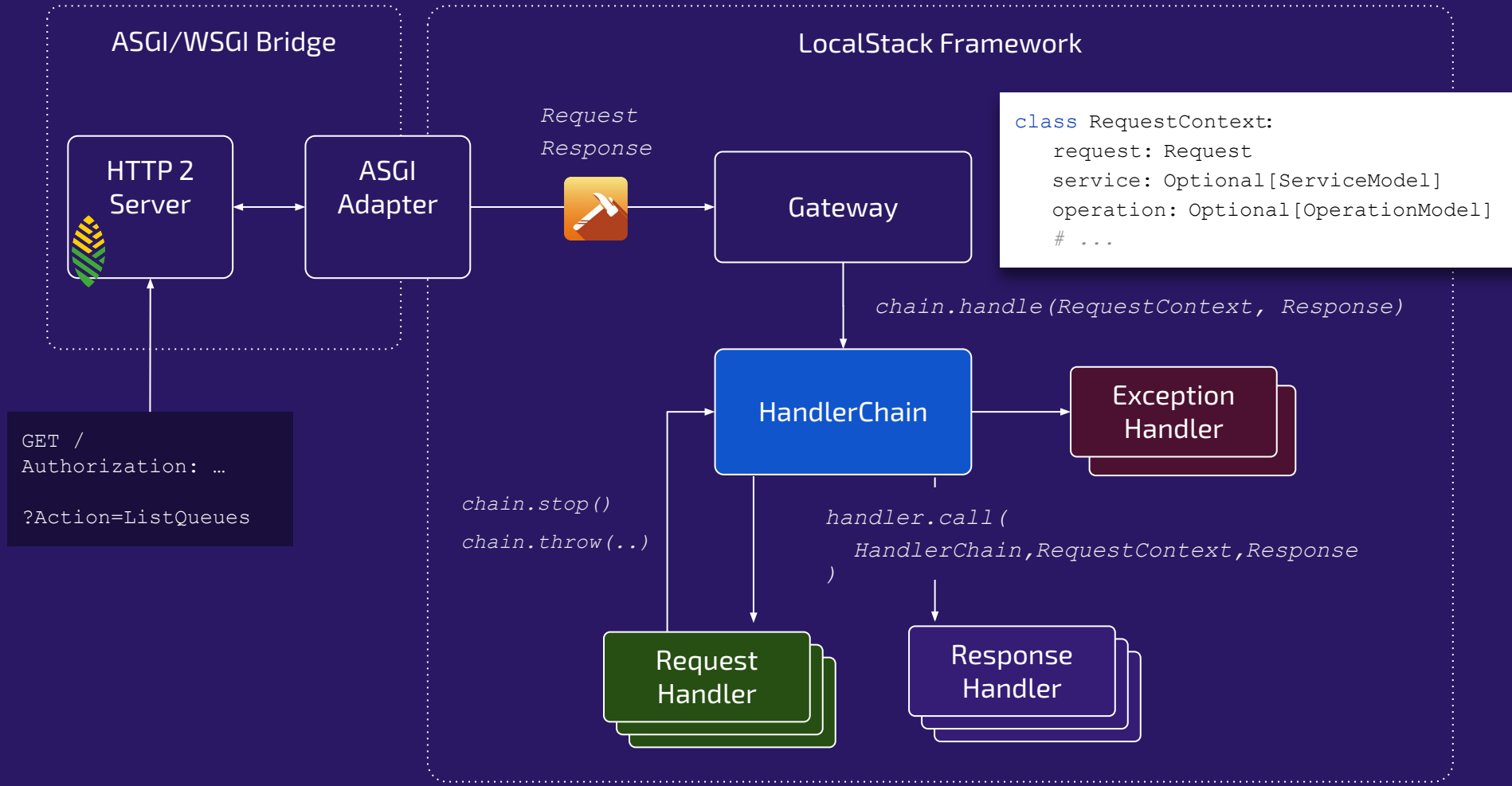
release v1.13 | Go v1.19 | license Apache-2.0

The Lambda Runtime Interface Emulator is a proxy for Lambda's Runtime and Extensions APIs, which allows customers to locally test their Lambda function packaged as a container image.

AWS Lambda Base Container Images

AWS provided base images for Lambda contain all the required components to run your functions packaged as container images on AWS Lambda. These base images contain the Amazon Linux Base operating system, the

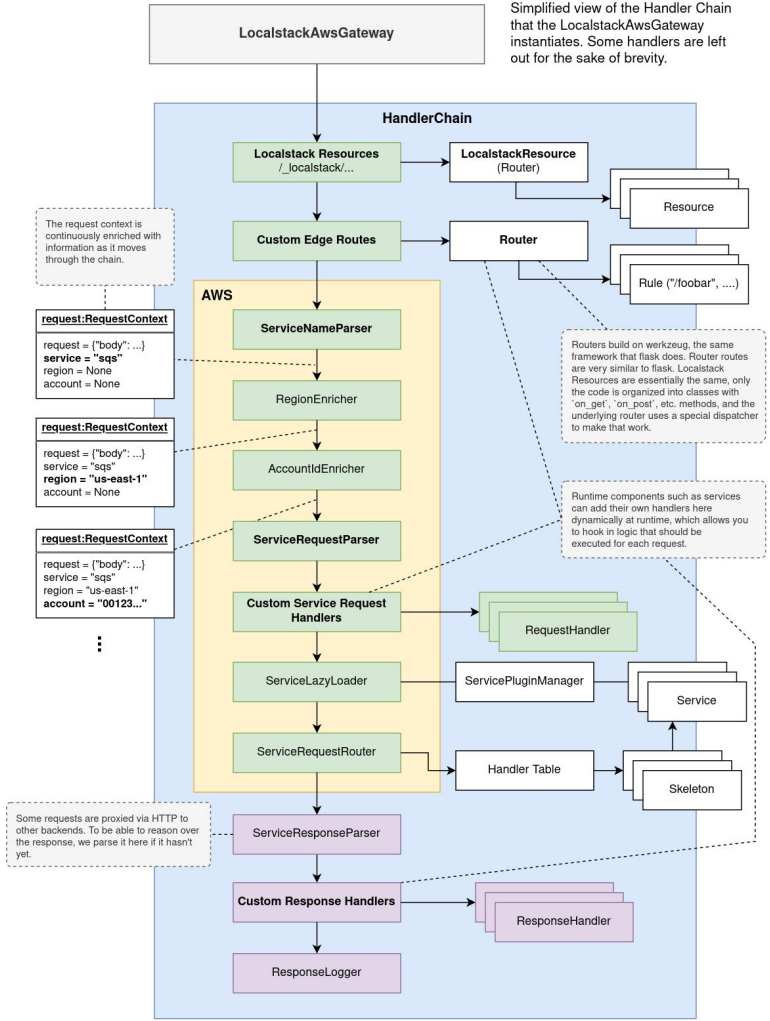
LocalStack's Web Application Framework


ASGI/WSGI Bridge
LocalStack Framework
HTTP 2 Server
ASGI Adapter
Request
Response
Gateway
class RequestContext:
request: Request
service: Optional[ServiceModel]
operation: Optional[OperationModel]
# ...
chain.handle(RequestContext, Response)
HandlerChain
Exception Handler
GET /
Authorization: …
?Action=ListQueues
chain.stop()
chain.throw(..)
handler.call(
HandlerChain,RequestContext,Response
)
Request Handler
Response Handler

```python
class ServiceNameParser(Handler):

    def __call__(self, chain: HandlerChain, context: RequestContext, response: Response):
        service = determine_aws_service_name(context.request)

        if not service:
            return

        context.service = self.get_service_model(service)

    def get_service_model(self, service: str) -> ServiceModel:
        return load_service(service)
```

```python
class ServiceRequestParser(Handler):

    parsers: Dict[str, RequestParser]

    def __init__(self):
        self.parsers = dict()

    def   call  (self, chain: HandlerChain, context: RequestContext, response: Response):
        if not context.service:
            LOG.debug("no service set in context, skipping request parsing")
            return

        return self.parse_and_enrich(context)

    def get_parser(self, service: ServiceModel):
        name = service.service_name

        if name in self.parsers:
            return self.parsers[name]
```

```python
class LocalstackAwsGateway(Gateway):
    def __init__(self, service_manager: ServiceManager = None) -> None:
        super().__init__()

        self.service_manager = service_manager or ServicePluginManager()
        self.service_request_router = ServiceRequestRouter()
        # lazy-loads services into the router
        load_service = ServiceLoader()

        # the main request handler chain
        self.request_handlers.extend(
            [
                handlers.push_request_context,
                handlers.preprocess_request,
                handlers.parse_service_name,
                handlers.enforce_cors,
                handlers.content_decoder,
                handlers.serve_localstack_resources,
                handlers.serve_default_listeners,
                handlers.serve_edge_router_rules,
                # start aws handler chain
                handlers.inject_auth_header_if_missing,
                handlers.add_region_from_header,
                handlers.add_account_id,
                handlers.add_internal_request_params,
                handlers.parse_service_request,
                handlers.serve_custom_service_request_handlers,
                load_service,
                self.service_request_router,
                # if the chain is still running, set an empty response
                EmptyResponseHandler(404, b'{"message": "Not Found"}'),
            ]
        )

        # exception handlers in the chain
        self.exception_handlers.extend(
            [
                handlers.log_exception,
                handlers.handle_service_exception,
```



```python
class HandlerToListenerAdapter(ProxyListener):

    chain: HandlerChain

    def forwa
        conte
        conte

        respo
        self.

        retur
```

```python
class ListenerToHandlerAdapter(Handler):

    listener: ProxyListener

    def __call__(self, chain: HandlerChain, context: RequestContext, response: Response):
        request = context.request

        result = self.listener.forward_request(
            method=request.method,
            path=request.full_path if request.query_string else request.path,
            data=restore_payload(request),
            headers=request.headers,
        )

        if type(result) == int:
            chain.respond(status_code=result)
            return

        if isinstance(result, ...)
            # ...

        raise ValueError(f"cannot create response for result{result}")
```


ADAPTER WIN
EPIC FAILS

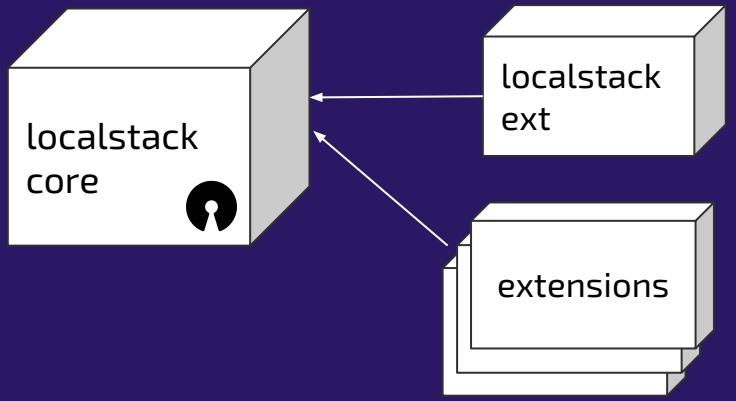
Pluggability

plugins.py

```
def do_register_localstack_plugins():
    # register default plugins
    try:
        from localstack.services import edge
        from localstack.services.acm import acm_starter
        from localstack.services.apigateway import apigateway_listener, apigateway_starter
        from localstack.services.cloudwatch import cloudwatch_listener, cloudwatch_starter
        from localstack.services.configservice import configservice_starter
        from localstack.services.dynamodb import dynamodb_listener, dynamodb_starter
        from localstack.services.ec2 import ec2_listener, ec2_starter
        from localstack.services.es import es_starter
        from localstack.services.events import events_listener, events_starter
        from localstack.services.iam import iam_listener, iam_starter
        from localstack.services.infra import (
            start_cloudformation,
            start_dynamodbstreams,
            start_firehose,
            start_lambda,
            start_sns,
            start_ssm,
            start_sts,
        )
        from localstack.services.kinesis import kinesis_listener, kinesis_starter
        from localstack.services.kms import kms_listener, kms_starter
        from localstack.services.logs import logs_listener, logs_starter
        from localstack.services.plugins import Plugin, register_plugin
        from localstack.services.redshift import redshift_starter
        from localstack.services.resourcegroups import rg_listener, rg_starter
        from localstack.services.resourcegroupstaggingapi import rgta_listener, rgta_starter
        from localstack.services.route53 import route53_listener, route53_starter
        from localstack.services.s3 import s3_listener, s3_starter
        from localstack.services.secretsmanager import (
            secretsmanager_listener,
            secretsmanager_starter,
        )
        from localstack.services.ses import ses_listener, ses_starter
        from localstack.services.sns import sns_listener
        from localstack.services.sqs import sqs_listener, sqs_starter
        from localstack.services.ssm import ssm_listener
        from localstack.services.stepfunctions import stepfunctions_listener, stepfunctions_starter
        from localstack.services.sts import sts_listener, sts_starter
        from localstack.services.support import support_starter
        from localstack.services.swf import swf_listener, swf_starter

        register_plugin(Plugin("edge", start=edge.start_edge, active=True))

        register_plugin(Plugin("acm", start=acm_starter.start_acm))

        register_plugin(
            Plugin(
```

import

- Took 5+ seconds
- No way to discover unknown packages
- … all the other obvious problems



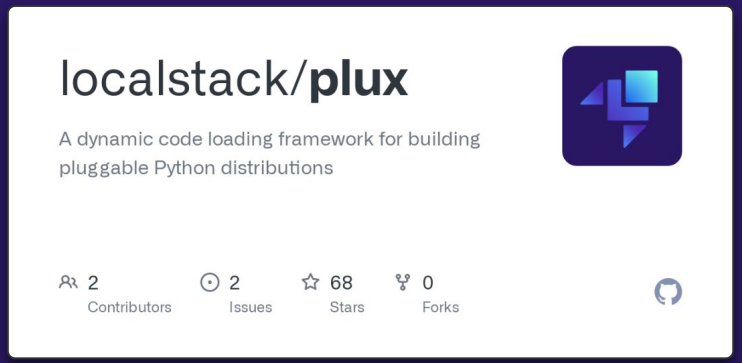

```python
from plugin import plugin

@plugin(namespace="localstack.configurators")
def configure_logging(runtime):
    logging.basicConfig(level=runtime.config.loglevel)

@plugin(namespace="localstack.configurators")
def configure_something(runtime):
    # defer imports to here and do something
```

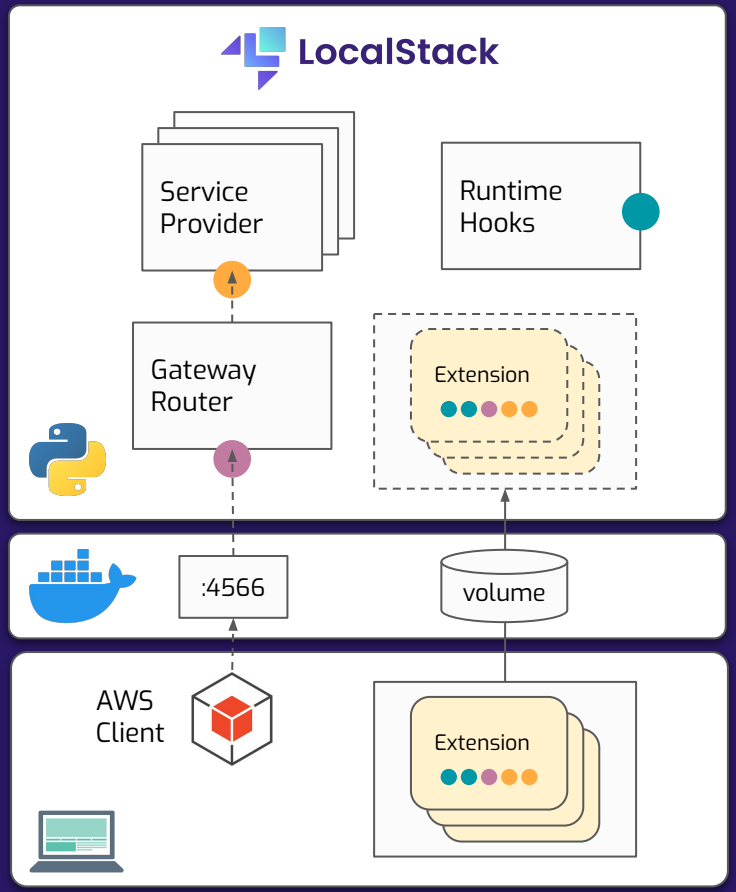
plux builds egg-info/entry_points.txt

```
[localstack.configurators]
configure_logging = localstack.plugins::configure_logging
configure_something = localstack.plugins::configure_something
```

plux loads entry points as plugins

```python
from plugin import PluginManager

plugins = PluginManager("localstack.configurators")
runtime = LocalstackRuntime()

for configurator in plugins.load_all():
    configurator(runtime)
```






```python
from localstack.extensions.api import Extension

class MyExtension(Extension):

    name = "my-extension"

    def on_platform_start(self):
        # run when localstack starts up
        pass

    def on_platform_ready(self):
        # run when localstack runtime is ready
        pass

    def update_gateway_routes(self, router: Router):
        # add additional HTTP routes
        pass

    def update_request_handlers(self, handlers: ...):
        # add AWS request handlers
        pass

    def update_response_handlers(self, handlers: ...):
        # add AWS response handlers
        pass
```

LocalStack: Turbocharging dev loops and team collaboration for cloud applications by Waldemar Hummer

Runtime Code Patching

- LocalStack uses a variety of third-party libraries
- In some cases, we need to slightly modify the existing logic of a function
  - Applying "**monkey patching**" - calling the original function, then adding some additional logic
- Ended up being a very common pattern
- Introduced a `@patch` decorator
  - Works for module attributes, static class attributes, and instance attributes

```python
import foo

def bar_new():
    result = bar_original()
    result.field = "updated field"
    return result

bar_original = foo.bar
foo.bar = bar_new
```

```python
import foo

@patch(foo.bar)
def bar_new(bar_original):
    result = bar_original()
    result.field = "updated field"
    return result
```

EuroPython
DUBLIN & REMOTE 11-17 July, 2022

36:00 / 46:33

https://www.youtube.com/watch?v=YIkKsnxb9eE

Parity

https://localstack.cloud/blog/2022-08-04-parity-explained

NEWS

LocalStack and AWS Parity Explained

At LocalStack we are committed to constantly improve the cloud dev experience. Here is how our AWS Server Framework and a new snapshot testing framework help us to stay on top of AWS changes.

8 MIN READ — POSTED AUGUST 4, 2022 BY **STEFANIE** AND **DOMINIK** AND **THOMAS**

Parity testing

pytest

```python
def test_basic_invoke (
    lambda_client, create_lambda, snapshot
):
    # custom transformers

snapshot.add_transformer(snaps

    # predefined name
    fn_name = f"ls-fn-{short_u

    # create the function
    response = create_lambda( F
    # record the response as pa
    snapshot.match("lambda_cre

    # invoke function
    invoke_result = lambda_clie
        FunctionName=fn_name,
        Payload=bytes("{}", "u
    )
    snapshot.match("lambda_invoke_result", invoke_result)
```

```
============================= FAILURES ==============================
_____________ TestLambdaAsfApi.test_basic_invoke ____________________
>> lambda_create_fn
        (-) /LastUpdateStatus
        (-) /Architectures
        (-) /VpcConfig
        (+) /MemorySize (128)
        (+) /EphemeralStorage ({'Size': 512})
        (+) /StateReasonCode (Creating)
        (+) /StateReason (The function is being created.)
        (~) /ResponseMetadata/HTTPStatusCode (200 → 201)
        (~) /State (Active → Pending)
```

Cloud Provider

```
{
 "test_lambda_api.py::TestLambda::test_basic_invoke": {
   "recorded-date": ...,
   "recorded-content": {
     "lambda_create_fn": {
       "ResponseMetadata": {
         "HTTPStatusCode": 201,
         "HTTPHeaders": {}
       },
       "FunctionName": "<function-name:1>",
                                    aws:lambda:<region>:111111111111:function:<function-nam
                                    9",
                                    ::111111111111:role/<resource:1>",
                                    ndler",
                                    e",
                                    J0J/jyyHt1fYZUuOqZ/Gc9Gm64Wp8fT6XNiXro=",
                                    gh"
                                    :1>",
                                    function is being created.",
       "StateReasonCode": "Creating",
       "PackageType": "Zip",
       "Architectures": [
         "x86_64"
       ],
       "EphemeralStorage": {
         "Size": 512
       }
     },
     "lambda_invoke_result": {
```

https://docs.localstack.cloud/references/coverage/

LocalStack Coverage

Overview of the implemented AWS APIs in LocalStack

Search for Service Name ...

ACM (AWS Certificate Manager)
Implementation details for API acm

Amplify
Implementation details for API amplify

API Gateway
Implementation details for API apigateway

API Gateway Management API
Implementation details for API apigatewaymanagementapi

API Gateway v2
Implementation details for API apigatewayv2

AppConfig
Implementation details for API appconfig

Application Auto Scaling
Implementation details for API application-autoscaling

AppSync
Implementation details for API appsync

Athena
Implementation details for API athena

Auto Scaling
Implementation details for API autoscaling

Backup
Implementation details for API backup

Batch
Implementation details for API batch

CE (Cost Explorer API)
Implementation details for API ce

CloudFormation
Implementation details for API cloudformation

CloudFront
Implementation details for API cloudfront

CloudTrail
Implementation details for API cloudtrail

CloudWatch
Implementation details for API cloudwatch

CodeCommit
Implementation details for API codecommit

Cognito Identity
Implementation details for API cognito-identity

Cognito IDP (Cognito User Pools API)
Implementation details for API cognito-idp

Config
Implementation details for API config

DocumentDB
Implementation details for API docdb

DynamoDB
Implementation details for API dynamodb

DynamoDB Streams
Implementation details for API

References / LocalStack Coverage / lambda

Lambda

Implementation details for API lambda

Coverage Overview

Lambda is supported by LocalStack in the **community ve**

Operation	Availal Implemented
AddLayerVersionPermission	✓
AddPermission	✓
CreateAlias	✓
CreateCodeSigningConfig	✓
CreateEventSourceMapping	✓
CreateFunction	✓
CreateFunctionUrlConfig	✓
DeleteAlias	✓
DeleteCodeSigningConfig	✓
DeleteEventSourceMapping	✓
DeleteFunction	✓
DeleteFunctionCodeSigningConfig	✓

CreateFunction

Parameters: Architectures, Code, Environment, FunctionName, Handler, Layers, Role, Runtime, Tags, Timeout

- LocalStack Pro
  - ✓ test_external_layer_specific HTTP Status Code: 201 AWS validated Snapshot Tested

Parameters: Architectures, Code, Environment, FunctionName, Handler, Role, Runtime, Tags, Timeout

- LocalStack Community
  - ✓ test_ignore_architecture HTTP Status Code: 201 AWS validated Snapshot Tested
  - ✓ test_runtime_introspection_x86 HTTP Status Code: 201 AWS validated Snapshot Tested

Parameters: Architectures, Code, FunctionName, Handler, PackageType, Role, Runtime

- LocalStack Community
  - ✓ test_create_lambda_exceptions HTTP Status Code: 400 (ValidationException) AWS validated Snapshot Tested
  - ✓ test_create_lambda_exceptions HTTP Status Code: 400 (ValidationException) AWS validated Snapshot Tested

Parameters: Architectures, Code, FunctionName, Handler, Role, Runtime

- LocalStack Community
  - ✓ test_sam_policies HTTP Status Code: 201 AWS validated Snapshot Tested

Parameters: Code, DeadLetterConfig, Environment, FunctionName, Handler, Role, Runtime, Tags, Timeout

- LocalStack Community
  - ✓ test_dead_letter_queue HTTP Status Code: 201 AWS validated Snapshot Tested
  - ✓ test_sqs_queue_as_lambda_dead_letter_queue HTTP Status Code: 201 AWS validated Snapshot Tested
  - ✓ test_sns_topic_as_lambda_dead_letter_queue HTTP Status Code: 201 AWS validated Snapshot Tested

Parameters: Code, Description, Environment, FunctionName, Handler, Role, Runtime, Tags, Timeout

- LocalStack Community
  - ✓ test_delete_on_nonexisting_version HTTP Status Code: 201 AWS validated Snapshot Tested
  - ✓ test_get_function_wrong_region[delete_function] HTTP Status Code: 201 AWS validated Snapshot Tested
  - ✓ test_get_function_wrong_region[get_function] HTTP Status Code: 201 AWS validated Snapshot Tested
  - ✓ test_get_function_wrong_region[get_function_code_signing_config] HTTP Status Code: 201 AWS validated Snapshot Tested

Takeaways


Dynamic code loading
Scaling your team = scaling your codebase
OSS ❤️
Pick your battles
Consider patterns when migrating frameworks
Introspect


Thank You! ❤️

Thomas Rausch

@thrauat

thrau

/in/thrau

Get involved!

LocalStack

https://localstack.cloud