1. Lambda-to-Endpoint Mapping Strategy 🔴

Q: For my domain API, how should I map endpoints to Lambda functions?

- Example: For a Quote API with operations like `create-quote`, `get-quote`, `update-quote`, `confirm-quote` - do I create:
  - ONE Lambda that routes internally based on path/method?
  - SEPARATE Lambdas per operation (4 Lambdas)?
  - GROUPED by "hot path" vs "cold path" (2 Lambdas)?

Why critical: This affects your entire project structure, testing strategy, and deployment complexity.

---

2. PolicyCenter Integration Details 🔴

Q: How exactly do I call PolicyCenter (the backend system)?

- What's the dev environment endpoint URL?
- Authentication mechanism? (API key, OAuth, mTLS, IAM?)
- Do you have SDK/client library or raw HTTP calls with `requests/httpx`?
- Can I get sample request/response payloads?
- What's the expected response time? (impacts timeout settings)

Why critical: You cannot write Lambda code without knowing how to call the backend.

---

3. Error Response Standard Format 🔴

Q: What is the exact JSON schema for error responses?

python

```
# Is it this?
{
 "error_code": "VALIDATION_ERROR",
 "message": "Invalid quote data",
 "timestamp": "2026-01-05T10:30:00Z"
}

# Or this?
{
 "statusCode": 400,
 "error": {"code": "VAL001", "detail": "..."}
}
```

Why critical: Pydantic models and error handling depend on this standard.

---

4. Stage Variables Configuration 🔴

Q: What stage variables are used and what are their exact names?

- Example needed: `POLICYCENTER_BASE_URL`, `TIMEOUT_MS`, `RETRY_COUNT`?
- How do I access them in Lambda? (via `event['stageVariables']`?)
- Can you share an example stage configuration?

Why critical: Your Lambda needs these to be environment-agnostic.

---

5. Lambda Versioning vs Aliases Strategy 🔴

Q: How do aliases work across stages?

- If `rlc3dev` stage uses `rlc3dev-alias` → version 5
- And `cupdev` stage uses `cupdev-alias` → version 3
- When I deploy version 6, do I manually update aliases via AWS Console or is there automation?
- How do I test version 6 in dev without affecting other dev stages?

Why critical: The document says "no API Gateway changes needed, just move alias pointer" but doesn't explain WHO moves it or HOW.

---

6. Testing Strategy Without Real Backend 🔴

Q: How do I test Lambda locally when PolicyCenter isn't available?

- Are there mock PolicyCenter endpoints in dev?
- Existing test fixtures/sample responses I should use?
- Can I use `moto` to mock AWS services + custom mocks for PolicyCenter?
- What's the minimum test coverage requirement before PR approval?

Why critical: You mentioned no Docker, only LocalStack/SAM. You need a clear mocking strategy.

---

7. S3 Lambda Package Bucket Details 🟡

Q: For Lambda deployment packages:

- What is the exact S3 bucket name format? (`dev-api-lambda-packages` or `<squad-name>-dev-api-lambda-packages`?)
- What is `ENV_ID` exactly? (account-level identifier or squad-specific?)
- Who manages this bucket? (DevOps team or do I create it?)
- How do I upload packages? (via CI/CD or manual `aws s3 cp`?)

Why critical: Impacts your deployment automation.

---

8. Secrets Manager Naming Convention 🟡

Q: For storing PolicyCenter credentials:

- What's the naming convention? (`/<env>/<domain>/policycenter/credentials`?)
- Example: `/dev/quote/policycenter/api-key`?
- Do secrets rotate? What's the rotation policy?
- How do I grant Lambda permission to read secrets? (IAM policy example needed)

Why critical: Security and configuration management dependency.

---

9. First Deliverable Scope 🔴

Q: What is the MVP for my first implementation?

- Specific domain? (Quote API, Policy API?)
- Specific endpoint? (POST /quote/v1/create-quote?)
- Expected timeline? (1 week, 2 weeks, 1 sprint?)
- Success criteria? (Working in dev? Deployed to test? Full observability setup?)

Why critical: Sets clear expectations and prevents scope creep.

---

10. CloudWatch Alarm Notification Channels 🟡

Q: When alarms trigger, where do they go?

- SNS topic ARN for dev environment?
- Teams channel webhook URL?
- Email distribution list?
- Who is responsible for responding to alarms in dev vs prod?
- Do I need to create SNS topics or are they pre-existing?

1. Pydantic Request/Response Models Structure 🔴

Q: What's the standard structure for request/response validation?

python

```
# Pattern A: Separate models per endpoint?
class CreateQuoteRequest(BaseModel):
    customer_id: str
    policy_type: str

class CreateQuoteResponse(BaseModel):
    quote_id: str
    premium: Decimal

# Pattern B: Shared base models with inheritance?
class QuoteBase(BaseModel):
    customer_id: str

class CreateQuoteRequest(QuoteBase):
    policy_type: str

# Pattern C: Nested models?
class QuoteRequest(BaseModel):
    data: CreateQuoteData
    metadata: RequestMetadata
```

- Do you have existing Pydantic model templates I should follow?
- Where do models live? (`src/models/`, `src/schemas/`?)
- Are there shared models across Lambda functions or per-Lambda models?

Why critical: Defines your entire data validation layer.

---

2. Lambda Handler Pattern & Routing 🔴

Q: How should I structure the Lambda handler for multiple endpoints?

python

```
# Pattern A: Single handler with internal routing?
def lambda_handler(event, context):
    route = event['path']
    method = event['httpMethod']
```

```
    if route == '/v1/quote/create' and method == 'POST':
        return handle_create_quote(event)
    elif route == '/v1/quote/get' and method == 'GET':
        return handle_get_quote(event)

# Pattern B: Use AWS Lambda Powertools Router?
from aws_lambda_powertools.event_handler import APIGatewayRestResolver

app = APIGatewayRestResolver()

@app.post("/v1/quote/create")
def create_quote():
    # handler logic
    pass

# Pattern C: Separate handler files per operation?
# quote_create.py, quote_get.py, etc.
```

- Which pattern is the team standard?
- Can you share an example Lambda handler from an existing API?

Why critical: Affects code organization, testability, and cold start performance.

---

3. PolicyCenter HTTP Client Implementation 🔴

Q: How do I make HTTP calls to PolicyCenter backend?

python

```
# Option A: Raw requests library?
import requests

response = requests.post(
    f"{POLICYCENTER_URL}/api/v1/quotes",
    json=payload,
    headers={"Authorization": f"Bearer {token}"},
    timeout=10
)

# Option B: httpx (async)?
```

```
import httpx

async with httpx.AsyncClient() as client:
    response = await client.post(...)

# Option C: boto3 if PolicyCenter has AWS API Gateway?
import boto3
client = boto3.client('execute-api')

# Option D: Custom SDK/client class?
from company.policycenter import PolicyCenterClient
client = PolicyCenterClient(base_url=..., auth=...)
```

- Is there an existing client class/wrapper I should use?
- Sync or async calls?
- Retry logic: Should I use `tenacity` library or manual retry?
- Timeout values: What should be the default timeout?

Why critical: Core business logic depends on this.

---

4. Error Handling & Exception Mapping 🔴

Q: How do I convert Python exceptions to API error responses?

python

```
# Pattern A: Custom exception classes?
class PolicyCenterError(Exception):
    def __init__(self, status_code: int, error_code: str, message: str):
        self.status_code = status_code
        self.error_code = error_code
        self.message = message

try:
    result = call_policycenter()
except requests.HTTPError as e:
    raise PolicyCenterError(502, "BACKEND_ERROR", str(e))

# Pattern B: Error response builder?
def build_error_response(status_code: int, error_code: str, message: str):
    return {
```

```
    'statusCode': status_code,
    'body': json.dumps({
      'error_code': error_code,
      'message': message,
      'timestamp': datetime.utcnow().isoformat()
    })
  }

# Pattern C: AWS Powertools error handling?
from aws_lambda_powertools.event_handler.exceptions import (
  BadRequestError,
  InternalServerError
)
```

- Do you have a standard exception hierarchy?
- Error response JSON schema? (see Question #3 from previous list)
- Should I catch all exceptions or let some bubble up?

Why critical: Consistent error handling across all APIs.

---

5. Logging Implementation with AWS Powertools 🔴

Q: What's the exact logging setup?

python

```
# Pattern A: AWS Powertools Logger?
from aws_lambda_powertools import Logger

logger = Logger(service="quote-api")

@logger.inject_lambda_context(log_event=True)
def lambda_handler(event, context):
  logger.info("Processing quote creation", extra={
    "quote_id": quote_id,
    "customer_id": customer_id
  })

# Pattern B: Standard logging with JSON formatter?
import logging
import json
```

```
logger = logging.getLogger()
logger.info(json.dumps({
   "level": "INFO",
   "message": "Processing quote",
   "quote_id": quote_id
}))

# Pattern C: Custom logger wrapper?
from company.logging import get_logger
logger = get_logger(__name__)
```

- **Required log fields:** You mentioned timestamp, log level, API name, stage, correlation ID, etc.
    - How do I access `stage` from Lambda? (from environment variable `STAGE` or event context?)
    - How do I extract/propagate correlation ID from API Gateway event?

python

```
# Example: Extracting correlation ID
correlation_id = event['headers'].get('X-Correlation-Id') or str(uuid.uuid4())
logger.append_keys(correlation_id=correlation_id)
```

- Is there a logging config file or all in code?

Why critical: Observability standard compliance (Section 5 of observability doc).

---

6. Environment Variable & Secrets Access Pattern 🔴

Q: How do I read configuration in Lambda?

python

```
# Pattern A: Direct environment variables?
import os

POLICYCENTER_URL = os.environ['POLICYCENTER_BASE_URL']
TIMEOUT_MS = int(os.environ.get('TIMEOUT_MS', '10000'))

# Pattern B: AWS Powertools Parameters?
from aws_lambda_powertools.utilities import parameters
```

```python
# SSM Parameter Store
policycenter_url = parameters.get_parameter("/dev/quote/policycenter/url")

# Secrets Manager
api_key = parameters.get_secret("dev/quote/policycenter/credentials")

# Pattern C: Pydantic Settings?
from pydantic_settings import BaseSettings

class Settings(BaseSettings):
    policycenter_base_url: str
    timeout_ms: int = 10000
    stage: str

    class Config:
        env_file = ".env"  # For local testing

settings = Settings()

# Pattern D: Custom config class?
from company.config import get_config
config = get_config(stage=os.environ['STAGE'])
```

- Which pattern is standard?
- Do secrets get cached? (AWS Powertools supports caching)
- How do I handle secret rotation during Lambda execution?

Why critical: Configuration management affects all Lambda functions.

---

7. AWS Powertools Features Usage 🟡

Q: Which AWS Powertools features should I use?

python

```
from aws_lambda_powertools import Logger, Tracer, Metrics
from aws_lambda_powertools.utilities.typing import LambdaContext
from aws_lambda_powertools.utilities.data_classes import APIGatewayProxyEvent
from aws_lambda_powertools.event_handler import APIGatewayRestResolver

# Tracer for X-Ray?
```

```
tracer = Tracer(service="quote-api")

# Metrics for custom CloudWatch metrics?
metrics = Metrics(namespace="QuoteAPI", service="quote-api")

@tracer.capture_lambda_handler
@logger.inject_lambda_context
@metrics.log_metrics(capture_cold_start_metric=True)
def lambda_handler(event: dict, context: LambdaContext):
    # Add custom metric
    metrics.add_metric(name="QuoteCreated", unit="Count", value=1)
```

- Are all features mandatory? (Tracer, Logger, Metrics, Validator, Parameters?)
- Are there team conventions for Powertools usage?
- Example Lambda using Powertools from existing codebase?

Why critical: Consistency with observability standards.

---

8. Testing Strategy & Mocking 🔴

Q: How do I write tests without real AWS services or PolicyCenter?

python

```
# Unit test with moto for AWS mocking
import pytest
from moto import mock_ssm, mock_secretsmanager
from src.handler import lambda_handler

@mock_ssm
@mock_secretsmanager
def test_create_quote():
    # Setup mocked AWS resources
    import boto3
    ssm = boto3.client('ssm', region_name='us-east-1')
    ssm.put_parameter(Name='/dev/quote/policycenter/url', Value='http://mock')

    # Mock PolicyCenter HTTP call
    with requests_mock.Mocker() as m:
        m.post('http://mock/api/v1/quotes', json={'quote_id': '123'})
```

```
    # Test Lambda handler
    event = {...}
    response = lambda_handler(event, {})

    assert response['statusCode'] == 200

# Integration test with real LocalStack?
@pytest.mark.integration
def test_create_quote_localstack():
    # Uses LocalStack endpoints
    pass
```

- Do you have a `conftest.py` with shared fixtures?
- Mock patterns for PolicyCenter responses?
- Minimum test coverage percentage? (80%? 90%?)
- Should I write integration tests or just unit tests?

Why critical: You need tests before PR approval.

---

9. **Directory Structure & File Organization** 🔴

Q: What's the standard Lambda project structure?

```
Option A: Flat structure
lambda_function/
├── handler.py        # Lambda handler
├── models.py         # Pydantic models
├── client.py         # PolicyCenter client
├── utils.py          # Helpers
└── requirements.txt

Option B: Layered structure
lambda_function/
├── src/
│   ├── handlers/
│   │   ├── __init__.py
│   │   ├── create_quote.py
```

```
│   │   └── get_quote.py
│   ├── models/
│   │   ├── __init__.py
│   │   ├── request.py
│   │   └── response.py
│   ├── services/
│   │   ├── __init__.py
│   │   └── policycenter_client.py
│   └── utils/
│       ├── __init__.py
│       ├── error_handler.py
│       └── logger.py
├── tests/
│   ├── unit/
│   └── integration/
├── pyproject.toml
└── README.md
```

Option C: Domain-driven structure

```
lambda_function/
├── domain/
│   ├── quote/
│   │   ├── handler.py
│   │   ├── models.py
│   │   └── service.py
├── infrastructure/
│   ├── clients/
│   └── config/
└── tests/
```

- Is there a cookiecutter template or scaffold script?
- Where does `pyproject.toml` live?
- Separate `tests/` folder or co-located with code?

Why critical: Affects code organization and imports.

---

10. API Gateway Event Parsing 🟡

Q: How do I extract data from API Gateway proxy event?

python

```python
# Manual parsing?
def lambda_handler(event, context):
    body = json.loads(event['body'])
    path_params = event['pathParameters']
    query_params = event['queryStringParameters']
    headers = event['headers']
    stage = event['requestContext']['stage']
    correlation_id = headers.get('X-Correlation-Id')

# AWS Powertools data classes?
from aws_lambda_powertools.utilities.data_classes import APIGatewayProxyEvent

def lambda_handler(event: dict, context):
    event_obj = APIGatewayProxyEvent(event)
    body = event_obj.json_body  # Automatically parsed
    correlation_id = event_obj.get_header_value('X-Correlation-Id')
    stage = event_obj.request_context.stage

# Pydantic validation on event?
class LambdaEvent(BaseModel):
    body: str
    headers: dict
    pathParameters: dict | None = None

    @property
    def parsed_body(self):
        return json.loads(self.body)
```

- Which approach is standard?
- How do I handle missing fields gracefully?

Why critical: Every Lambda needs to parse API Gateway events.

Bonus: Code Review Questions

11. Type Hints & mypy Configuration 🟢

Q: What's the type checking standard?

python

```python
# Strict typing required?
def create_quote(request: CreateQuoteRequest) -> CreateQuoteResponse:
    pass

# Optional typing?
def create_quote(request):
    pass
```

- Is mypy in strict mode?
- `mypy.ini` configuration file available?

---

12. Async vs Sync Lambda 🟢

Q: Should Lambda handlers be async?

python

```python
# Sync handler
def lambda_handler(event, context):
    return {'statusCode': 200}

# Async handler (requires Python 3.8+ runtime)
async def lambda_handler(event, context):
    result = await async_call()
    return {'statusCode': 200}
```

- Does PolicyCenter support async calls?
- Performance benefit worth the complexity?