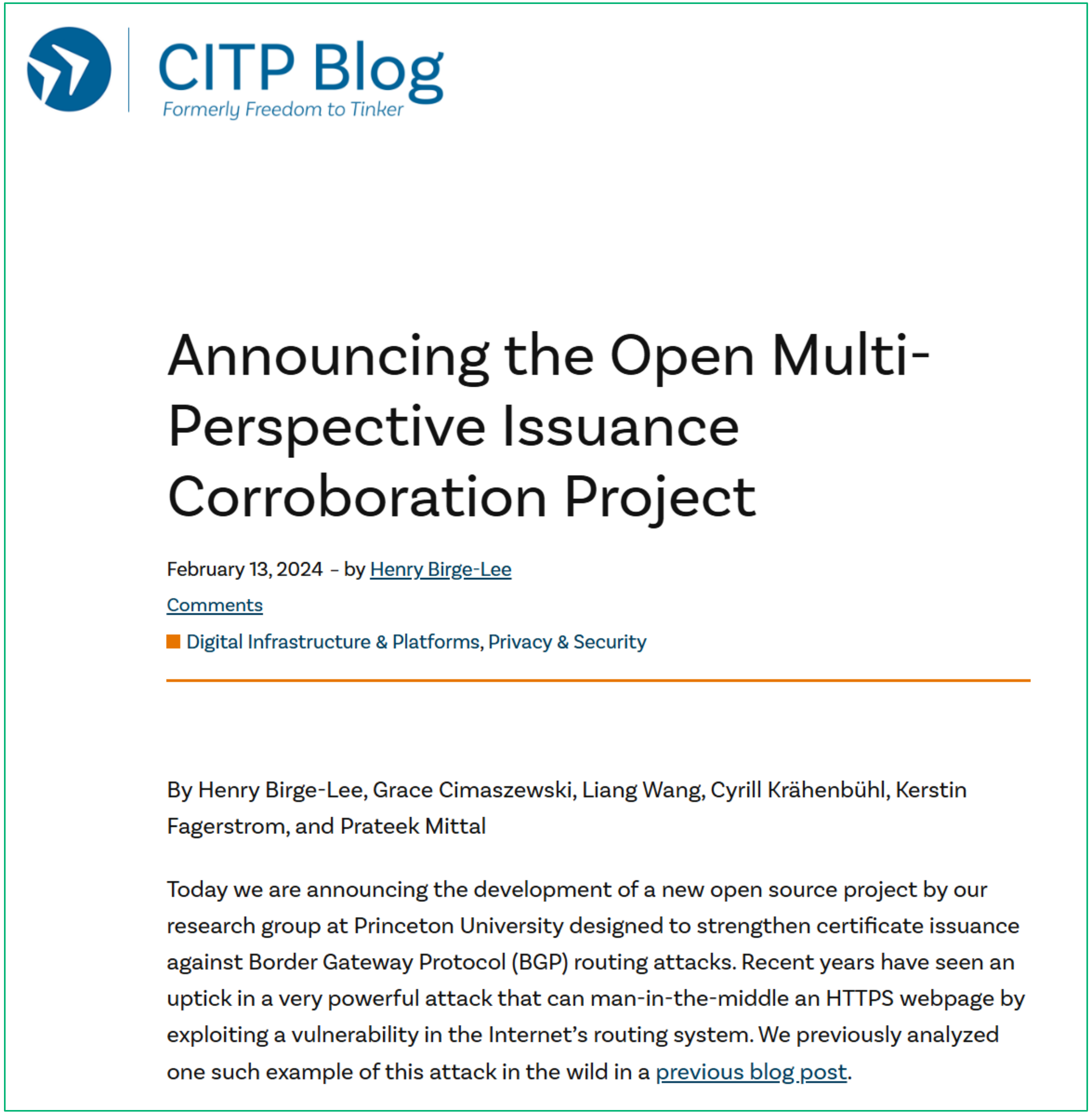
OPEN MPIC

A Turn-Key Solution to Protect Against BGP-Based Attacks

Dmitry Sharkov

March 2025

2/2024: Open MPIC Project Begins



CITP Blog

Formerly Freedom to Tinker

Announcing the Open Multi-Perspective Issuance Corroboration Project

February 13, 2024 – by Henry Birge-Lee

Comments

🟧 Digital Infrastructure & Platforms, Privacy & Security

By Henry Birge-Lee, Grace Cimaszewski, Liang Wang, Cyrill Krähenbühl, Kerstin Fagerstrom, and Prateek Mittal

Today we are announcing the development of a new open source project by our research group at Princeton University designed to strengthen certificate issuance against Border Gateway Protocol (BGP) routing attacks. Recent years have seen an uptick in a very powerful attack that can man-in-the-middle an HTTPS webpage by exploiting a vulnerability in the Internet's routing system. We previously analyzed one such example of this attack in the wild in a previous blog post.

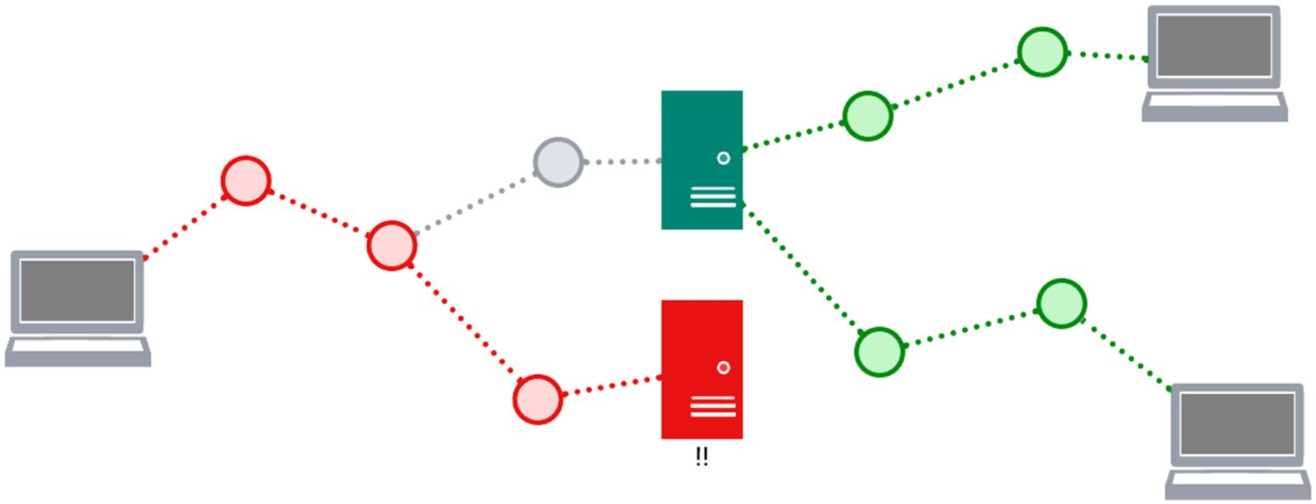
PRINCETON CS

SECURITY AND PRIVACY RESEARCH GROUP

Implications of MPIC Requirements

Functional Requirements

- Enforcing minimum quorum given number of perspectives
- Comprehensive, specific logging of results

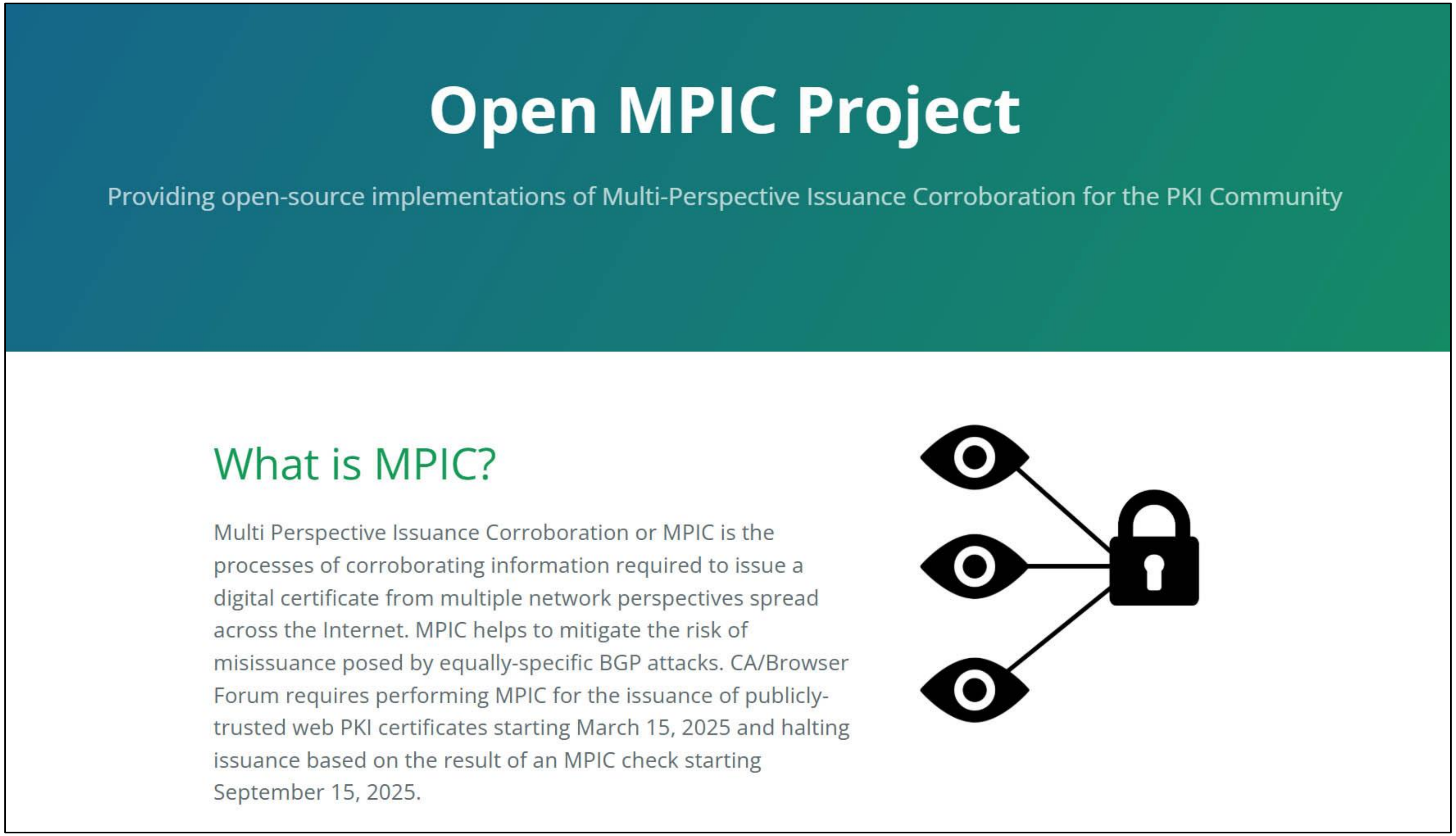


Implications of MPIC Requirements

Geographic Requirements

- Multiple RIRs per set of corroborating perspectives
- Minimum distance of 500km between perspectives

open-mpic.org



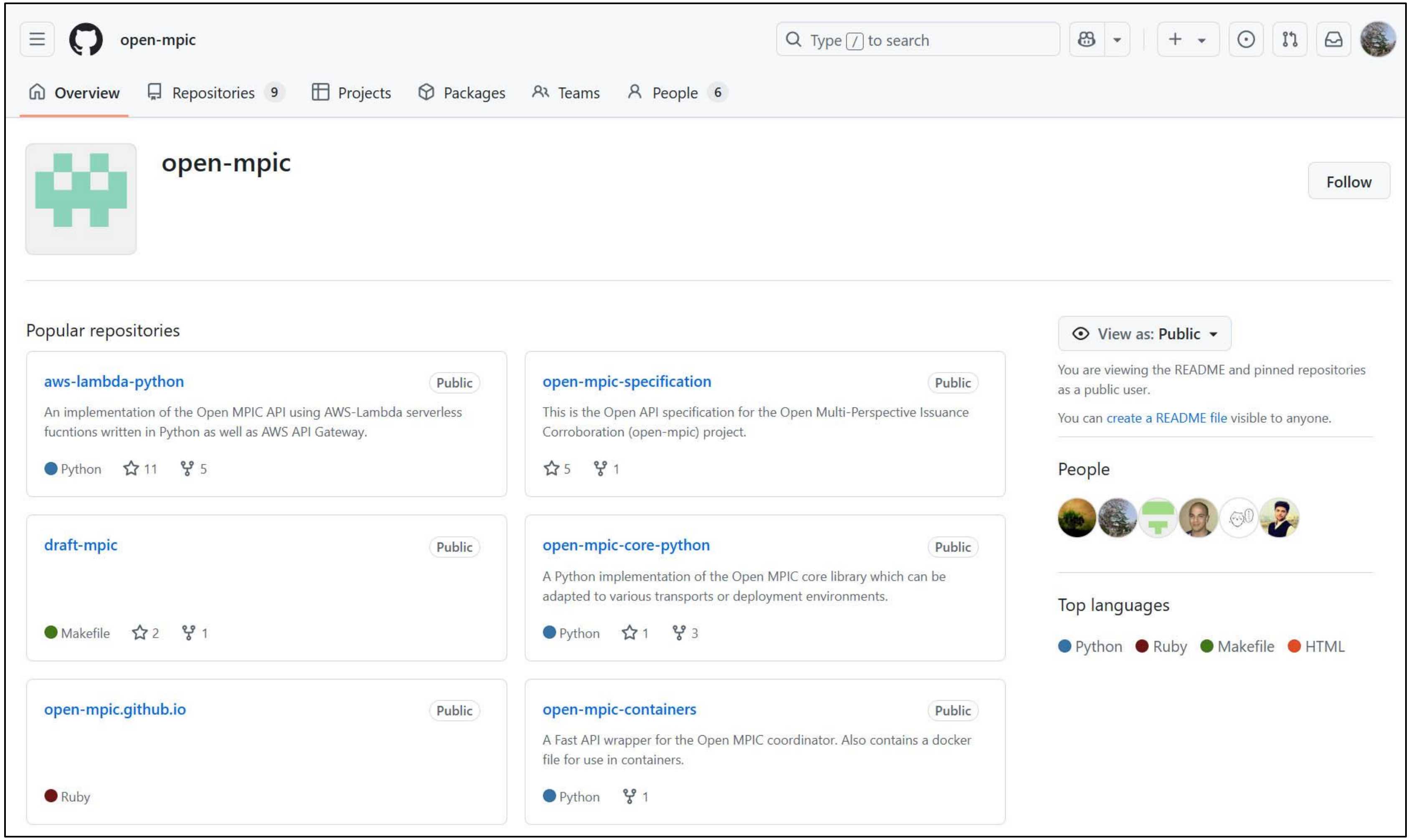
Open MPIC Project

Providing open-source implementations of Multi-Perspective Issuance Corroboration for the PKI Community

What is MPIC?

Multi Perspective Issuance Corroboration or MPIC is the processes of corroborating information required to issue a digital certificate from multiple network perspectives spread across the Internet. MPIC helps to mitigate the risk of misissuance posed by equally-specific BGP attacks. CA/Browser Forum requires performing MPIC for the issuance of publicly-trusted web PKI certificates starting March 15, 2025 and halting issuance based on the result of an MPIC check starting September 15, 2025.

github.com/open-mpic



open-mpic

Popular repositories

aws-lambda-python Public

An implementation of the Open MPIC API using AWS-Lambda serverless fucntions written in Python as well as AWS API Gateway.

● Python ☆ 11 Forks 5

open-mpic-specification Public

This is the Open API specification for the Open Multi-Perspective Issuance Corroboration (open-mpic) project.

☆ 5 Forks 1

draft-mpic Public

● Makefile ☆ 2 Forks 1

open-mpic-core-python Public

A Python implementation of the Open MPIC core library which can be adapted to various transports or deployment environments.

● Python ☆ 1 Forks 3

open-mpic.github.io Public

● Ruby

open-mpic-containers Public

A Fast API wrapper for the Open MPIC coordinator. Also contains a docker file for use in containers.

● Python Forks 1

You are viewing the README and pinned repositories as a public user.

You can create a README file visible to anyone.

People

Top languages

● Python ● Ruby ● Makefile ● HTML

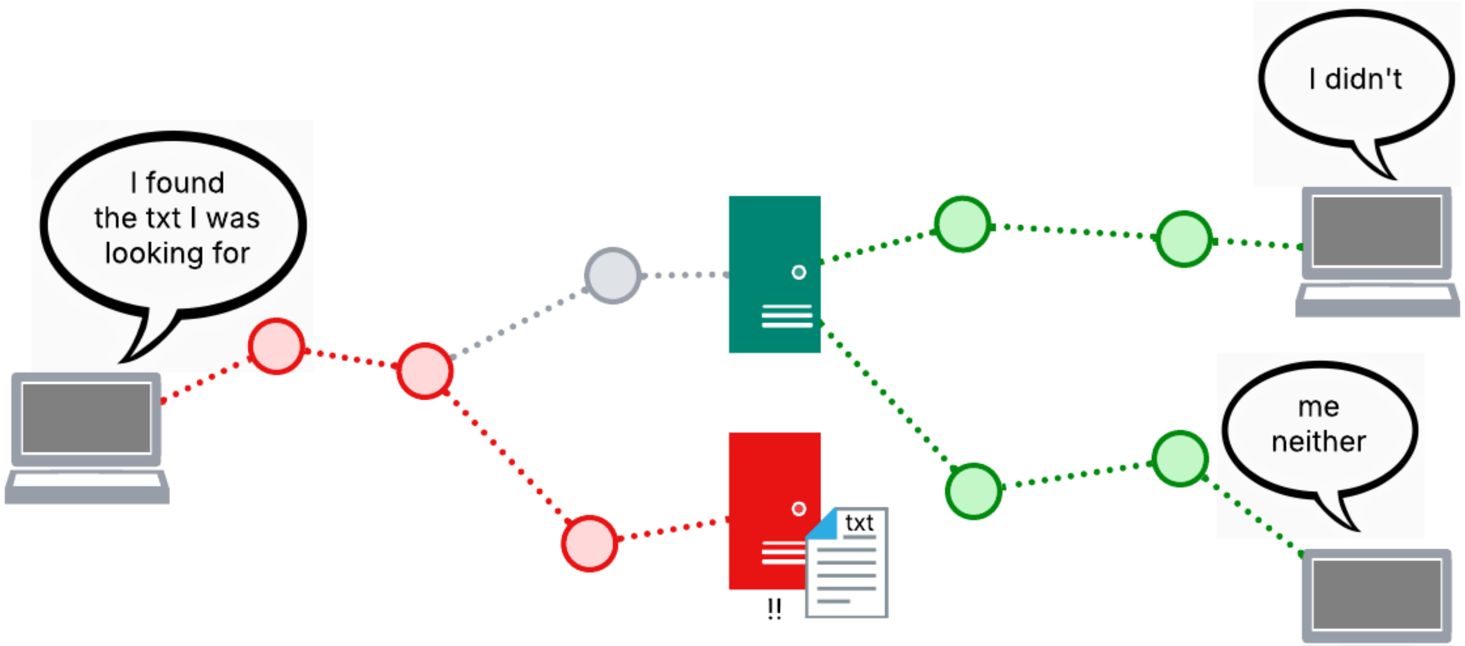
Open MPIC's Value Proposition

- One-size-fits-all
- Open source (MIT license)
- Stateless REST API
- ACME and non-ACME validation
- Self-hosted
- Paint-by-numbers deployment

This is important because DCV is important.

DCV **<u>must</u>** be trustworthy.

Or there is no *authority*
to a Certificate Authority.



JULY 2024

MPIC Project
open-source imp
rspective Issuance Corroboration for th
MPIC
SECTIGO®
PRINCETON
ENGINEERING

WHY IT MADE SENSE

SECTIGO®

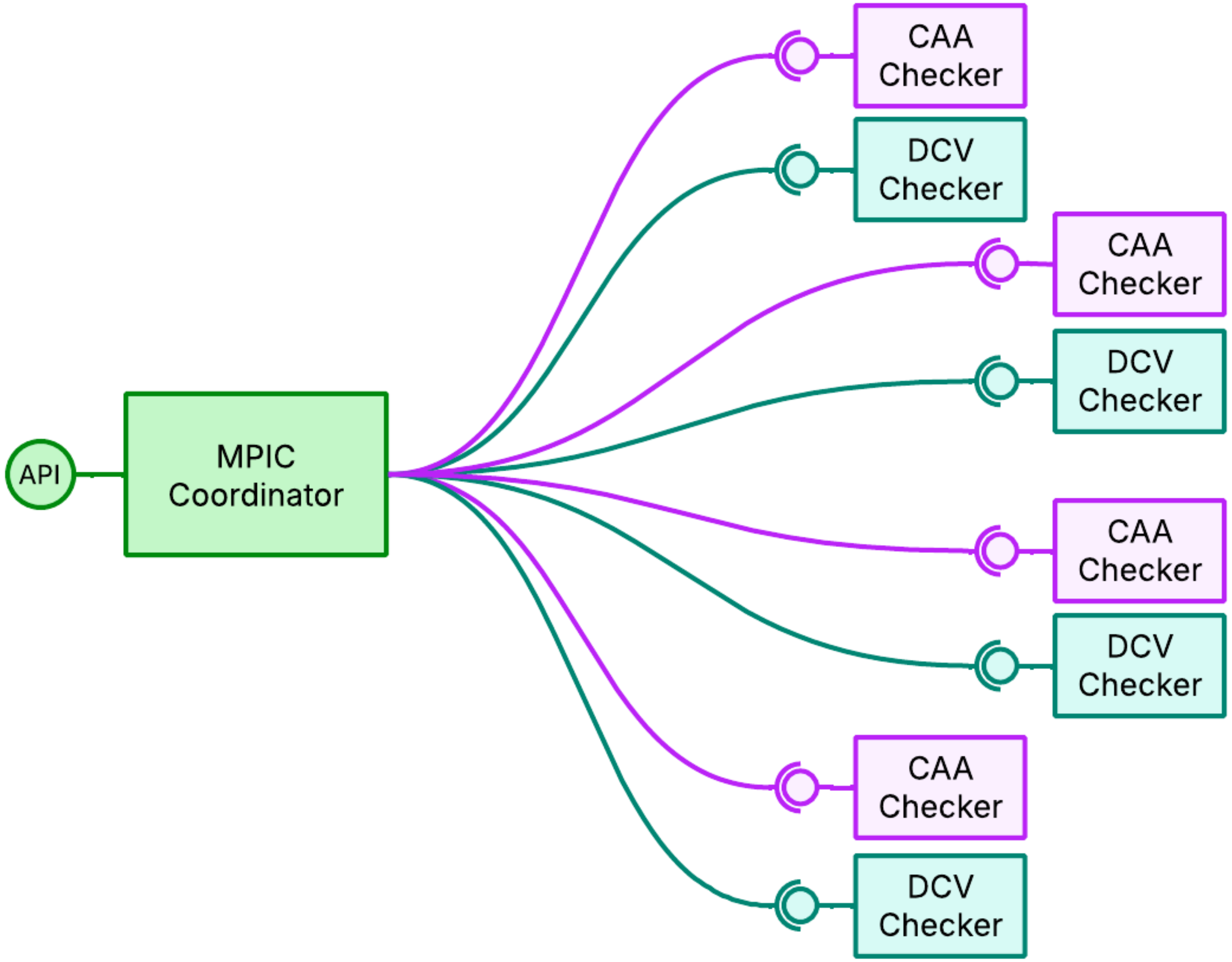
- Needed to implement MPIC in any case.
- Believes in stewardship of a secure Internet.
- Collaboration with MPIC's foremost experts.

PRINCETON ENGINEERING

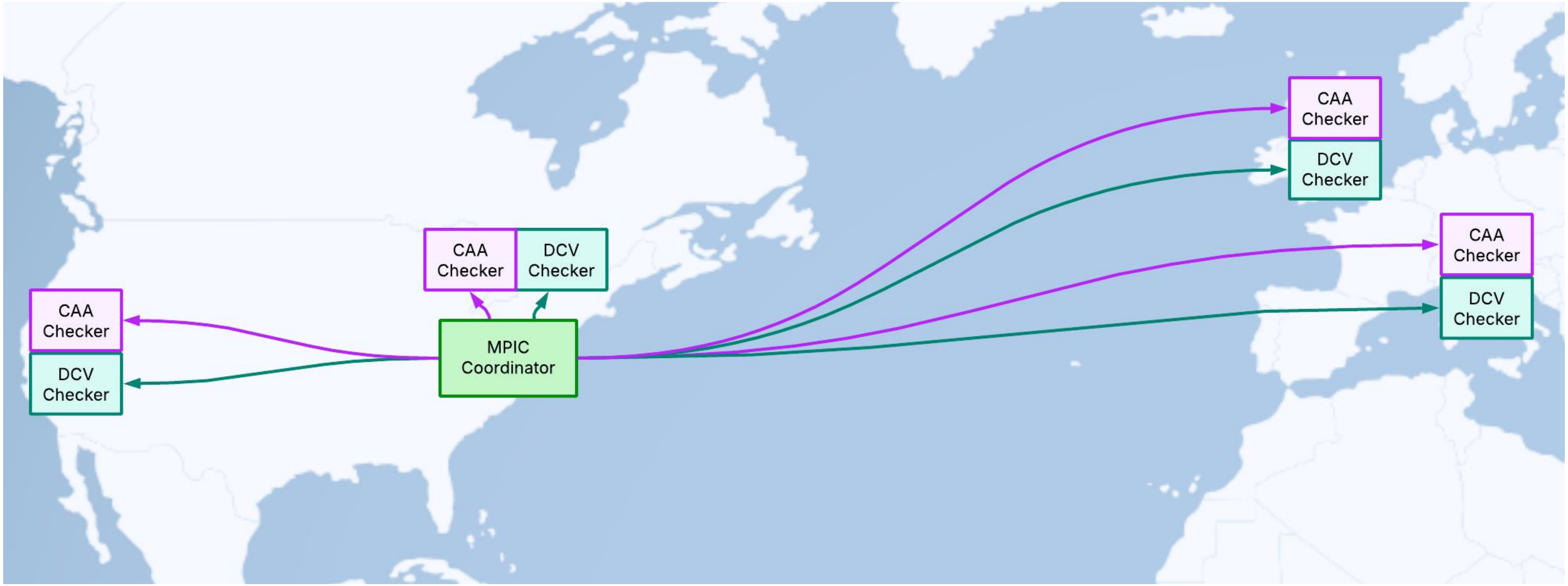
- Solution was at early proof-of-concept stage. Adoption was not assured.
- Believes in stewardship of a secure Internet.
- Collaboration with a large CA and engineering organization.

High Level Overview

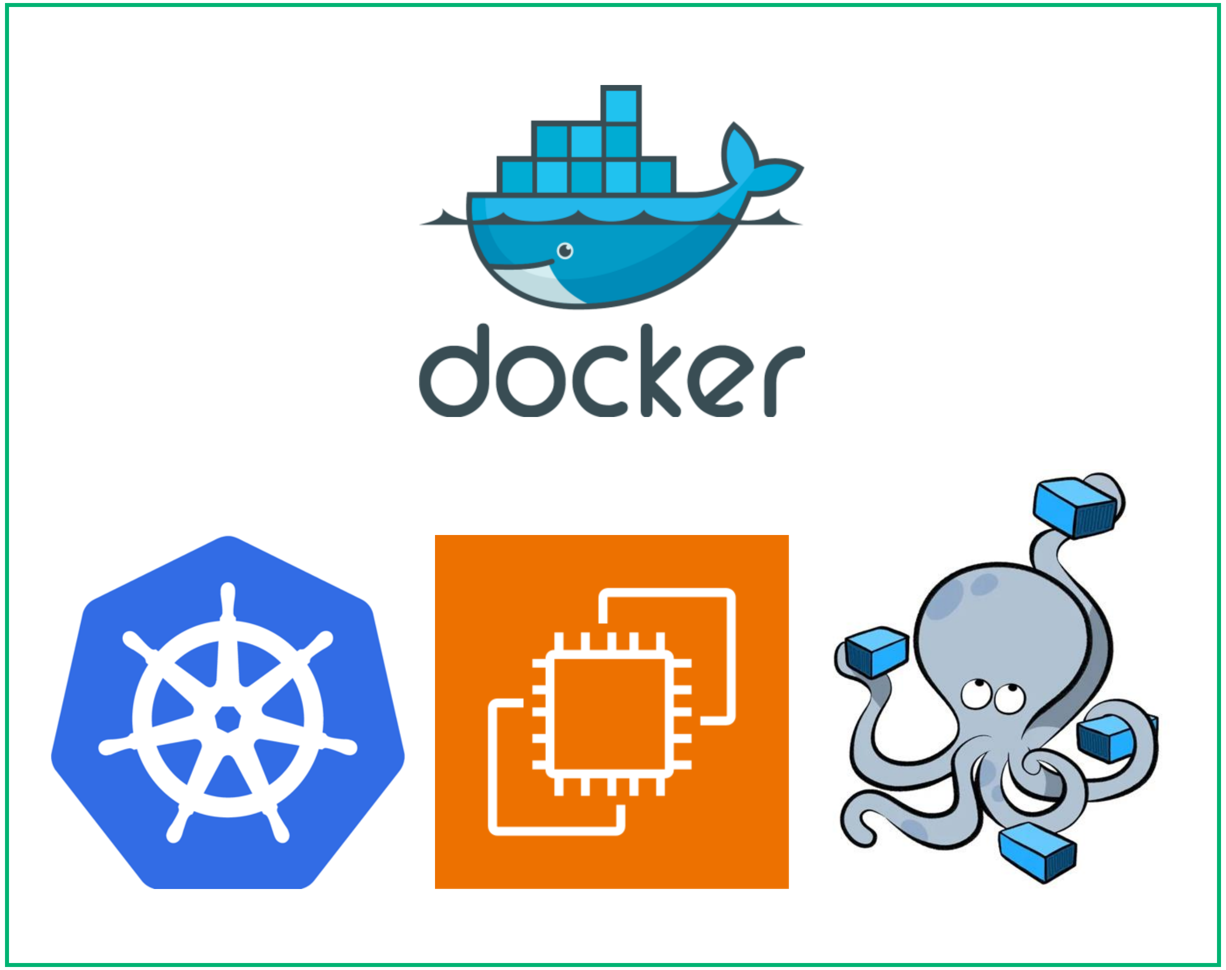
Open MPIC Topology



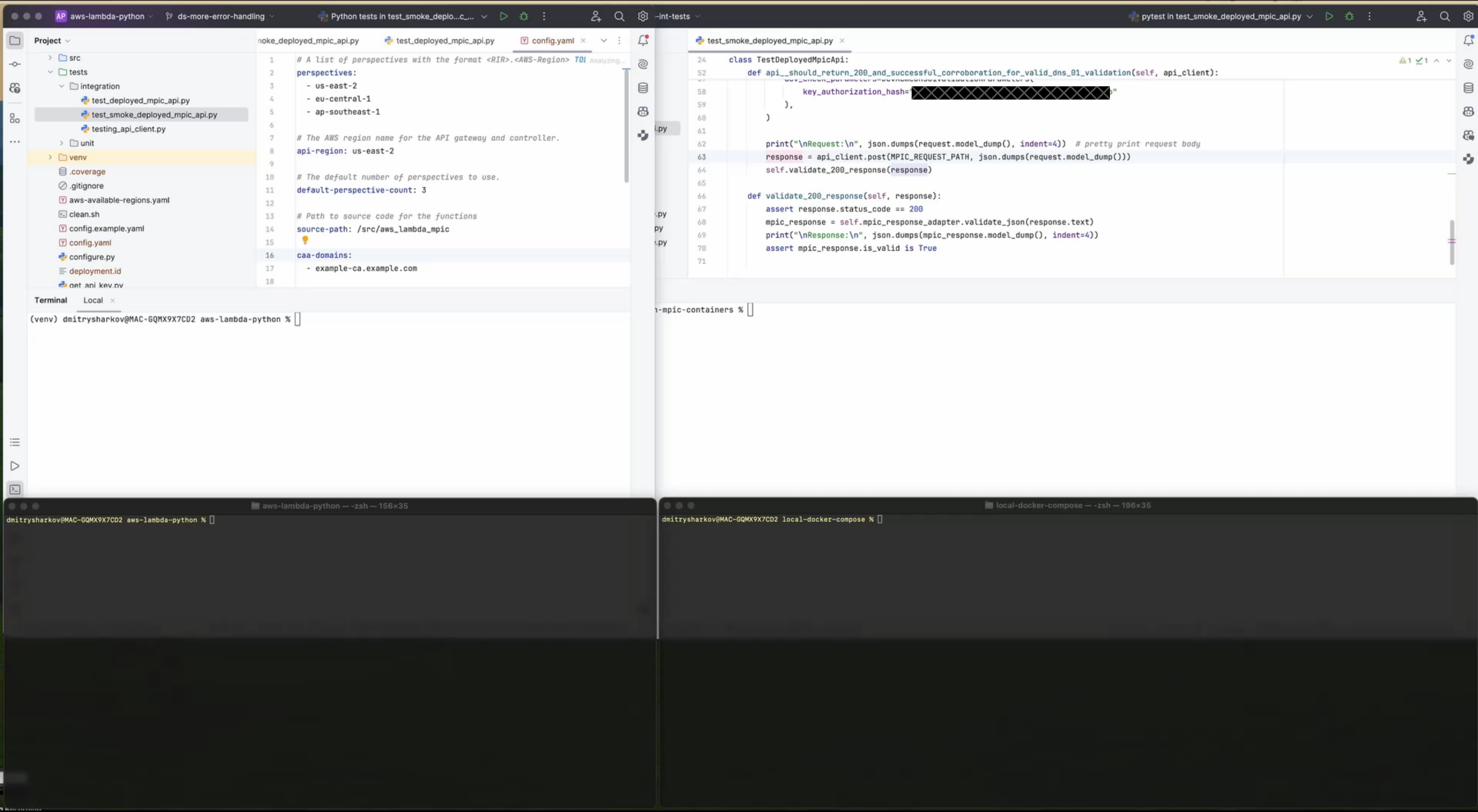
Open MPIC Topography



Deployment Options



DEMO


aws-lambda-python
ds-more-error-handling
Python tests in test_smoke_deplo...c...
Project
src
tests
integration
test_deployed_mpic_api.py
test_smoke_deployed_mpic_api.py
testing_api_client.py
unit
venv
.coverage
.gitignore
aws-available-regions.yaml
clean.sh
config.example.yaml
config.yaml
configure.py
deployment.id
get_api_key.py
noke_deployed_mpic_api.py
test_deployed_mpic_api.py
config.yaml
Terminal
Local
-int-tests
pytest in test_smoke_deployed_mpic_api.py
test_smoke_deployed_mpic_api.py
aws-lambda-python — -zsh — 156×35
local-docker-compose — -zsh — 196×35

```yaml
# A list of perspectives with the format <RIR>.<AWS-Region> TOI
perspectives:
  - us-east-2
  - eu-central-1
  - ap-southeast-1

# The AWS region name for the API gateway and controller.
api-region: us-east-2

# The default number of perspectives to use.
default-perspective-count: 3

# Path to source code for the functions
source-path: /src/aws_lambda_mpic


caa-domains:
  - example-ca.example.com

```

```
(venv) dmitrysharkov@MAC-GQMX9X7CD2 aws-lambda-python % 
```

```python
class TestDeployedMpicApi:
    def api__should_return_200_and_successful_corroboration_for_valid_dns_01_validation(self, api_client):
                    key_authorization_hash="[redacted]"
                ),
            )

        print("\nRequest:\n", json.dumps(request.model_dump(), indent=4))  # pretty print request body
        response = api_client.post(MPIC_REQUEST_PATH, json.dumps(request.model_dump()))
        self.validate_200_response(response)

    def validate_200_response(self, response):
        assert response.status_code == 200
        mpic_response = self.mpic_response_adapter.validate_json(response.text)
        print("\nResponse:\n", json.dumps(mpic_response.model_dump(), indent=4))
        assert mpic_response.is_valid is True

```

```
-mpic-containers % 
```

```
dmitrysharkov@MAC-GQMX9X7CD2 aws-lambda-python % 
```

```
dmitrysharkov@MAC-GQMX9X7CD2 local-docker-compose % 
```

Building Open MPIC

Functional Completeness

- Nearly all required DCV validation methods supported (and CAA of course).
- All data that must be persisted is returned through JSON payload.
- Logging and tracing for monitoring and observability.
- Request / configuration validation.

DCV METHOD	Open MPIC Supported
3.2.2.4.7 DNS Change	Yes
3.2.2.4.8 IP Address	Yes
3.2.2.4.13 Email to DNS CAA Contact	Yes
3.2.2.4.14 Email to DNS TXT Contact	Yes
3.2.2.4.16 Phone to DNS TXT Contact	Yes
3.2.2.4.17 Phone to DNS CAA Contact	Yes
3.2.2.4.18 Change to Website v2	Yes
3.2.2.4.19 Change to Website – ACME	Yes
3.2.2.4.20 TLS Using ALPN	No (yet)
3.2.2.5.1 Change to Website	Yes
3.2.2.5.3 Reverse Address Lookup	Yes
3.2.2.5.6 ACME “http-01” for IP	Yes
3.2.2.5.7 ACME “tls-alpn-01” for IP	Yes

Stability and Agility Through Testing

- Full rewrite, test-driven.
- Code not covered by tests is auto rejected.
- Robust continuous integration and delivery pipeline.
- Necessary for business-critical OSS.
- Enables making improvements, like new deployment options, quickly.

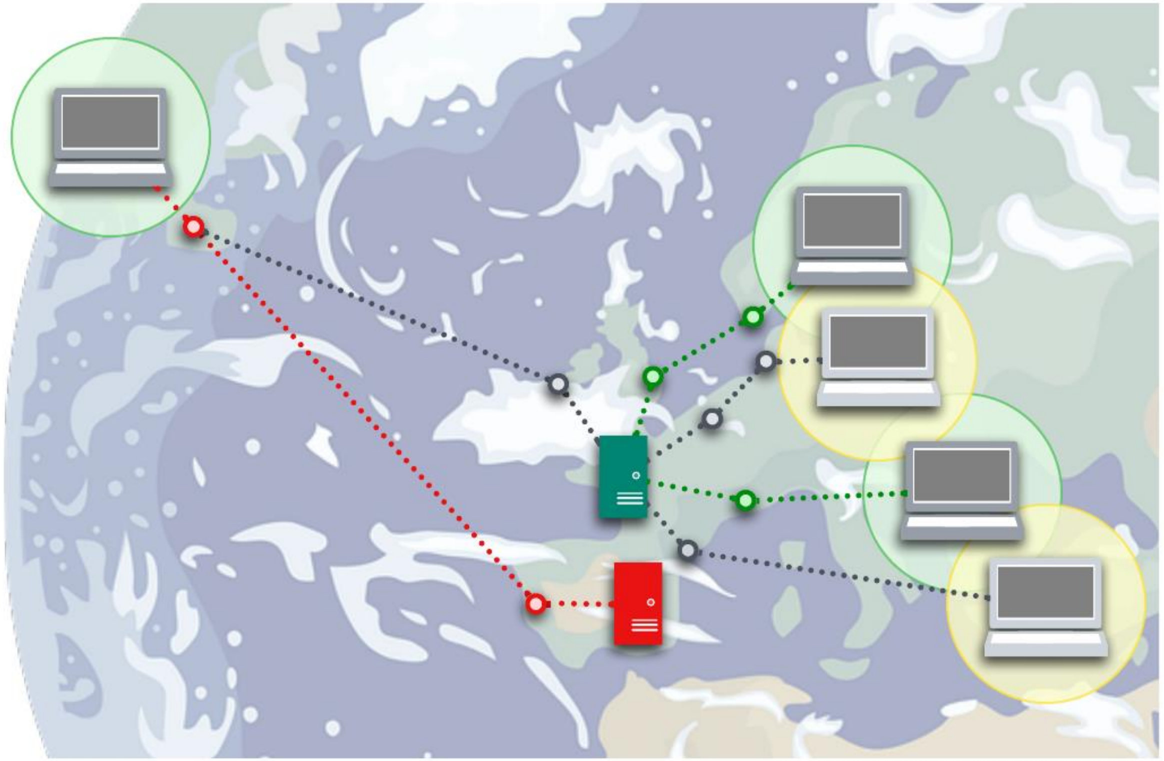
Tech Stack

Python	Pydantic
FastAPI	Uvicorn

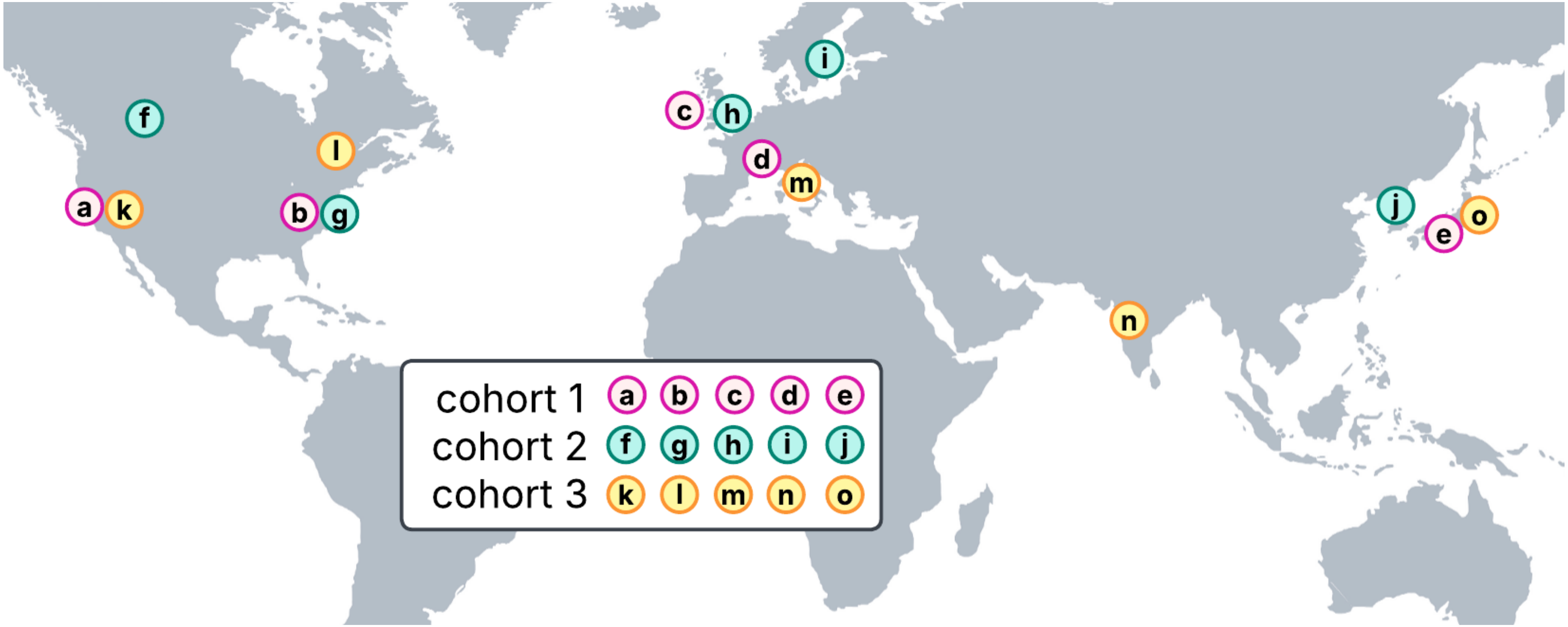
```
available_regions:
  -
    code: "ap-northeast-1"
    name: "Asia Pacific (Tokyo)"
    rir: "apnic"
    too_close_codes: ["ap-northeast-3"]
  -
    code: "ap-northeast-3"
    name: "Asia Pacific (Osaka)"
    rir: "apnic"
    too_close_codes: ["ap-northeast-1"]
  -
    code: "ca-central-1"
    name: "Canada (Central)"
    rir: "arin"
    too_close_codes: []
  -
    code: "ca-west-1"
    name: "Canada West (Calgary)"
    rir: "arin"
    too_close_codes: []
  -
    code: "eu-central-1"
    name: "Europe (Frankfurt)"
    rir: "ripe"
    too_close_codes: ["eu-central-2"]
  -
    code: "eu-central-2"
    name: "Europe (Zurich)"
    rir: "ripe"
    too_close_codes: ["eu-central-1", "eu-
south-1"]
```

Perspective Distances

- "Shared Responsibility Model" (user defines, service enforces)
- Allows for creating perspective "cohorts."

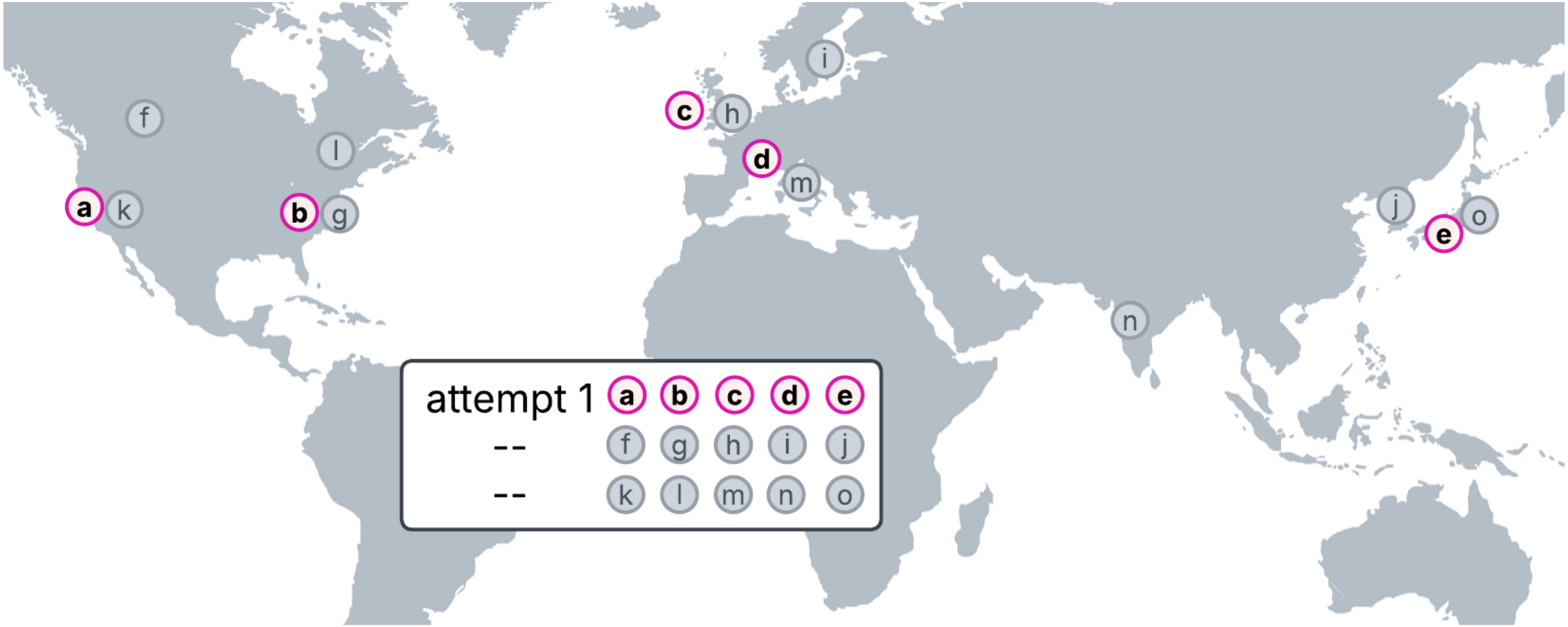


Retries



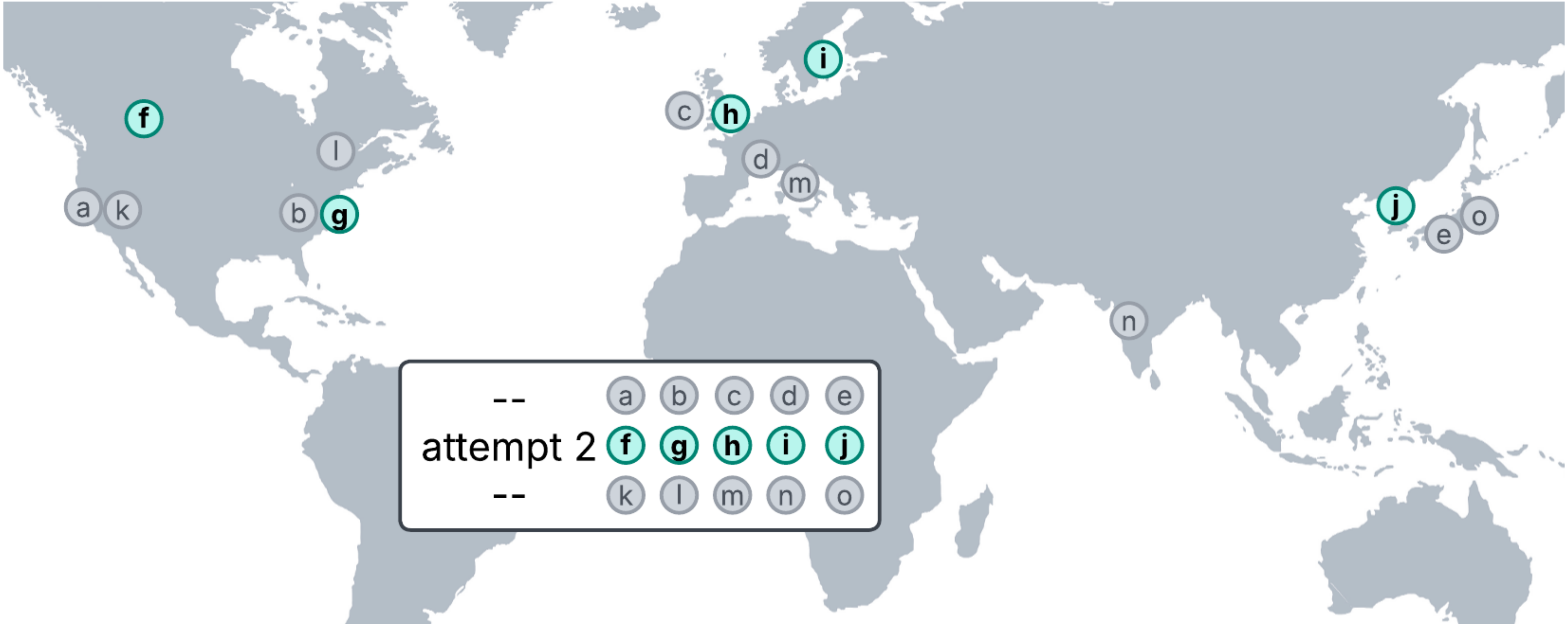
i
c
h
f
l
d
m
a
k
b
g
j
o
e
n
cohort 1 a b c d e
cohort 2 f g h i j
cohort 3 k l m n o

Retries

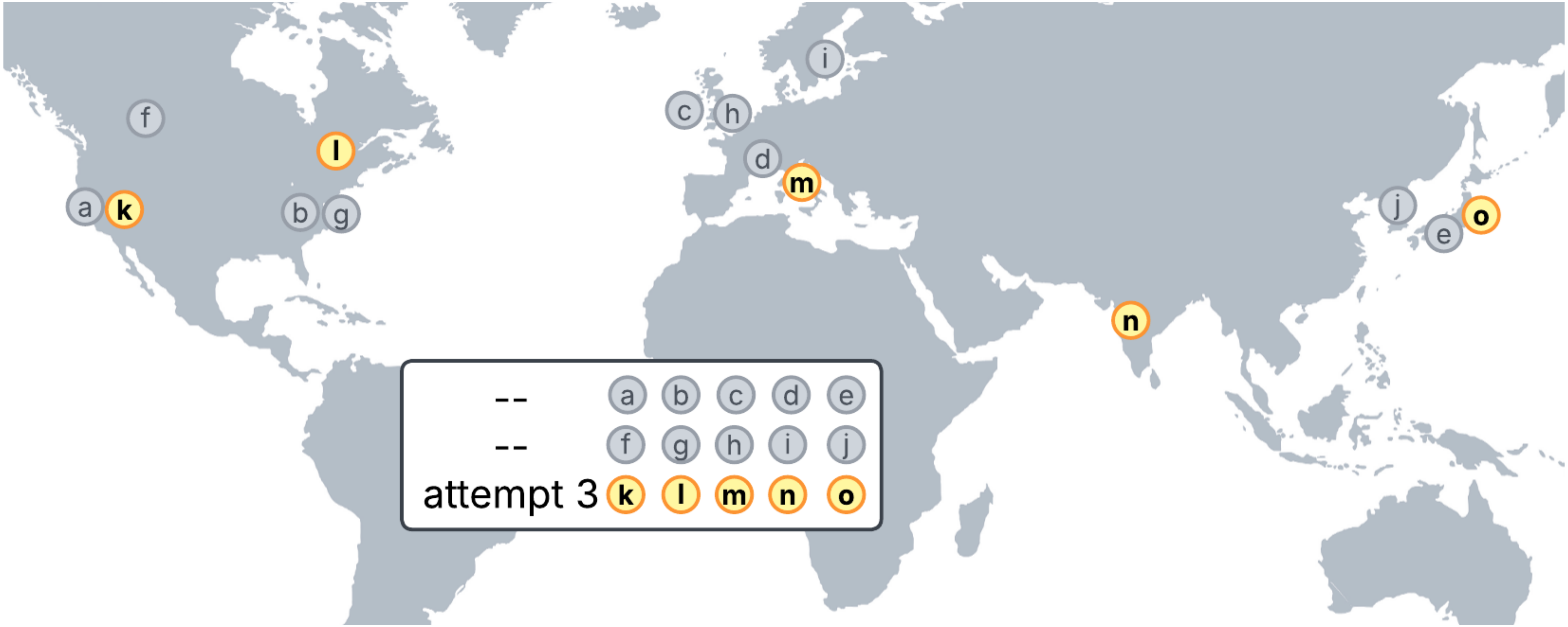


attempt 1	a	b	c	d	e
--	f	g	h	i	j
--	k	l	m	n	o

Retries



Retries



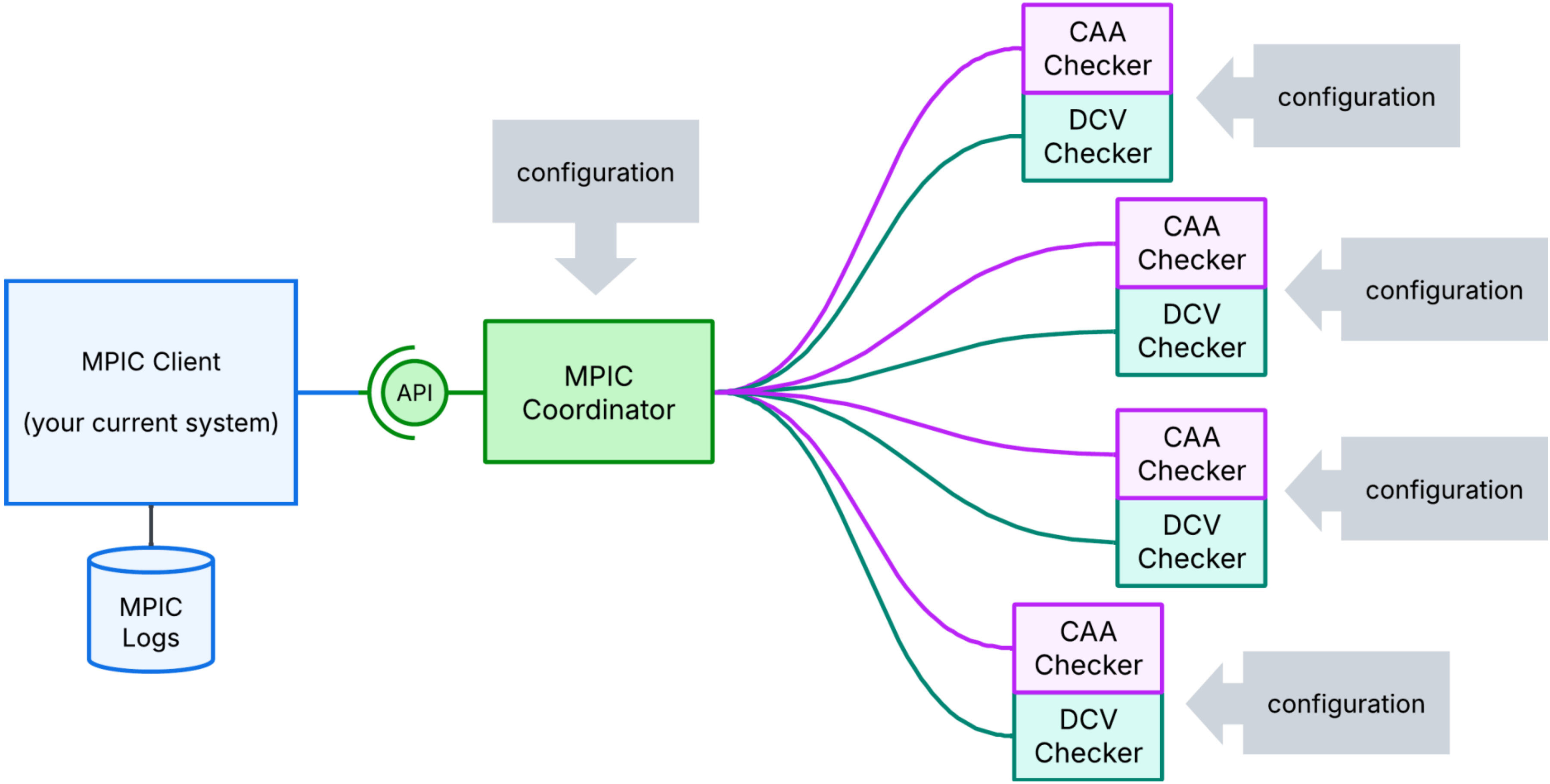
Using Open MPIC

Deploying Open MPIC

- Self-hosted – user downloads, builds, configures, and runs it.
- Fully stateless – provision more hardware to scale it horizontally (Lambda auto scales).
- *Unbound* DNS resolver container (with baseline configuration) included.

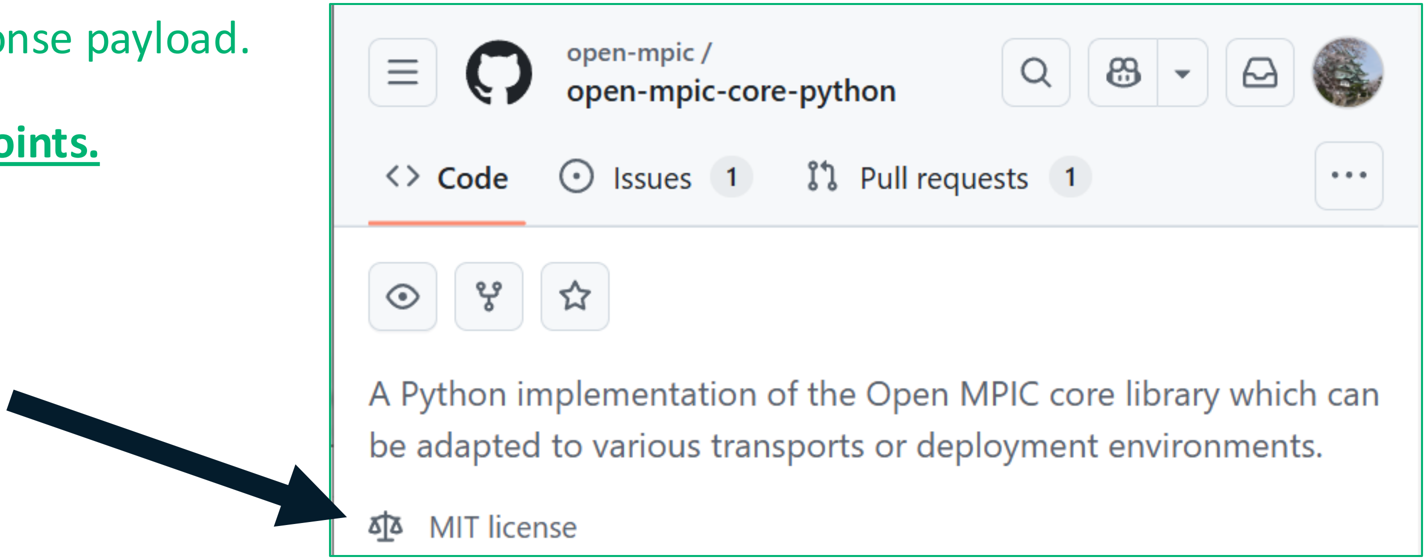
- Configurable:
  - Perspectives (locations)
  - Logging and log level
  - Timeouts
  - Retries
- AWS Lambda deployment requires an account and user with appropriate permissions.

Deploying Open MPIC



Who Does What?

- Open MPIC carries out the remote CAA and DCV checks.
- Open MPIC enforces certain requirements automatically, others based on configuration.
- CA needs to provide correct and valid configuration.
- Open MPIC automatically sorts perspectives into cohorts and performs retries as requested.
- Open MPIC returns a payload in a single JSON that contains everything to meet logging requirements.
- CA needs to successfully log the response payload.
- **CA needs to secure Open MPIC endpoints.**



open-mpic /
open-mpic-core-python

A Python implementation of the Open MPIC core library which can be adapted to various transports or deployment environments.

MIT license

Current State of Open MPIC

- Ready to use
- Currently deployed in production
- Officially maintained by SECTIGO®
- PRINCETON ENGINEERING acting as core maintainer and chief steward
- More partners are very welcome

Thank You

Dmitry Sharkov