Safely deploying a 100K line Envoy YAML configuration to production

Jyoti Mahapatra & Lisa Lu

Speakers

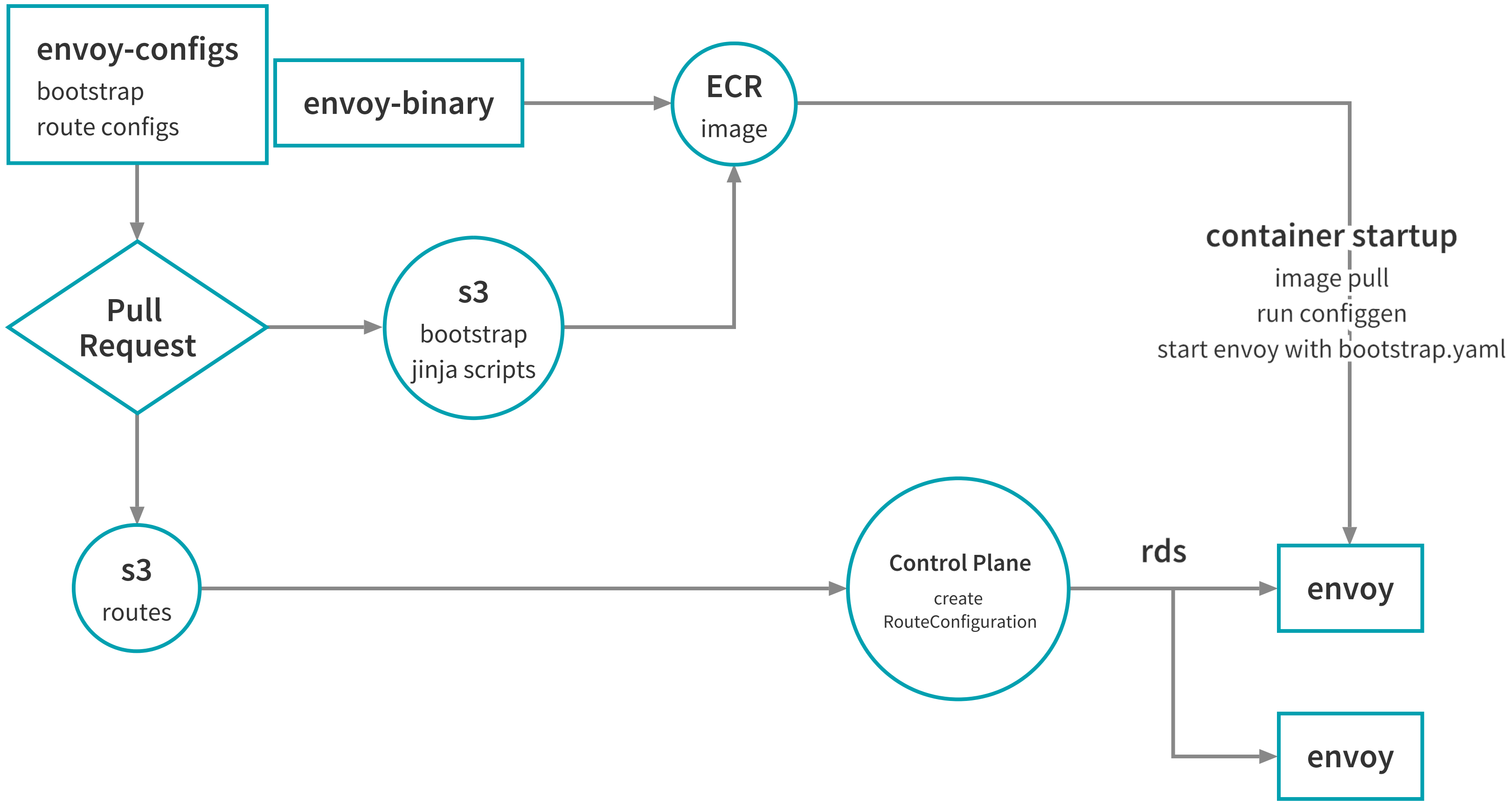
LisaLudique

seaweedfarmer

jyotimahapatra

jyotireloaded

xDS configuration infra



MADE WITH beautiful.ai

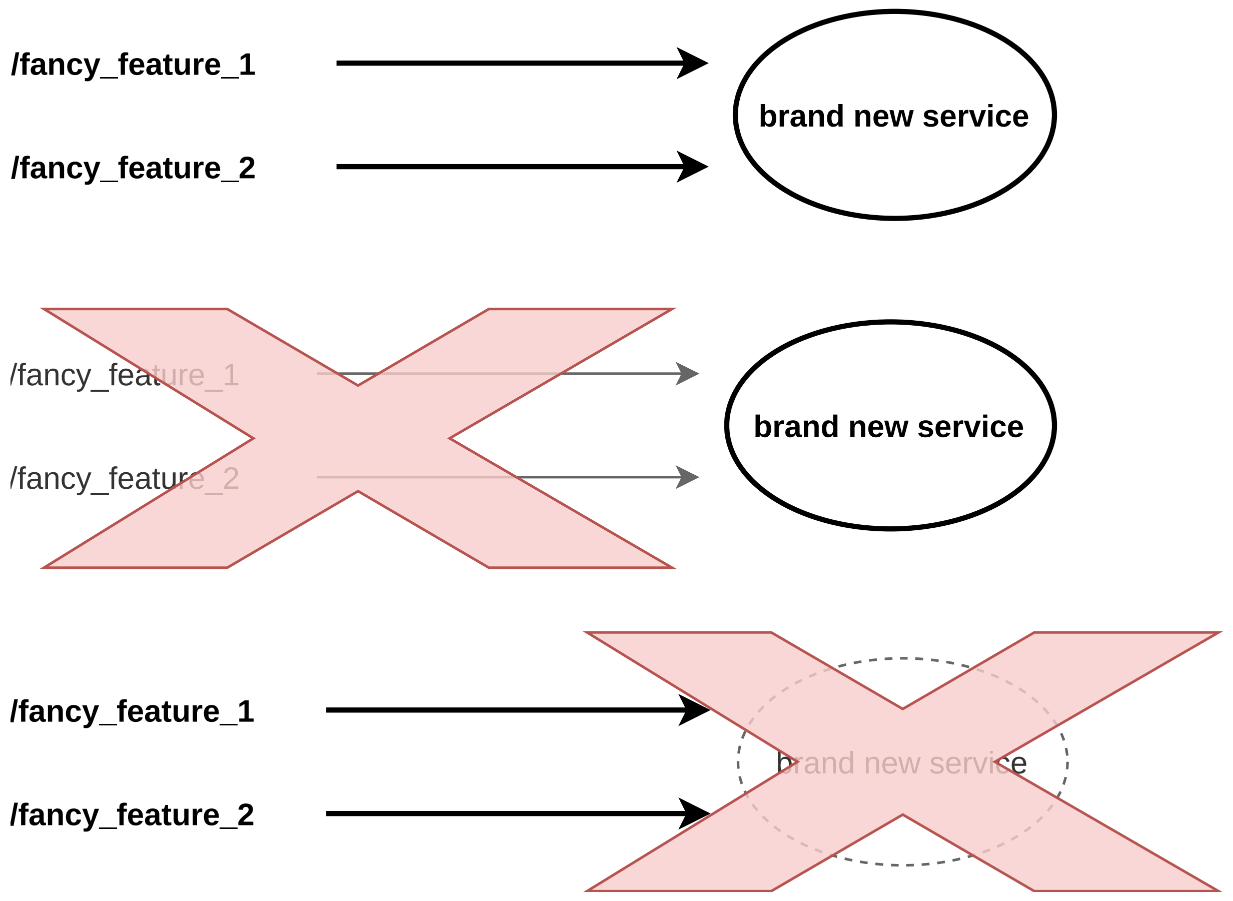
Problems

1. **Increased oncall burden**
   More PRs, more Slack pings, more tickets
2. **Human error in configs**
   Missed errors led to service disruptions
3. **Slow response time**
   Slow iteration and slow rollouts
4. **Tech debt**
   Dead routes and dead clusters piling up...
5. **Hard to test**
   End-to-end setup needed to validate changes
6. **Stability**
   Frequent envoy updates can bring in deprecated fields

Goal

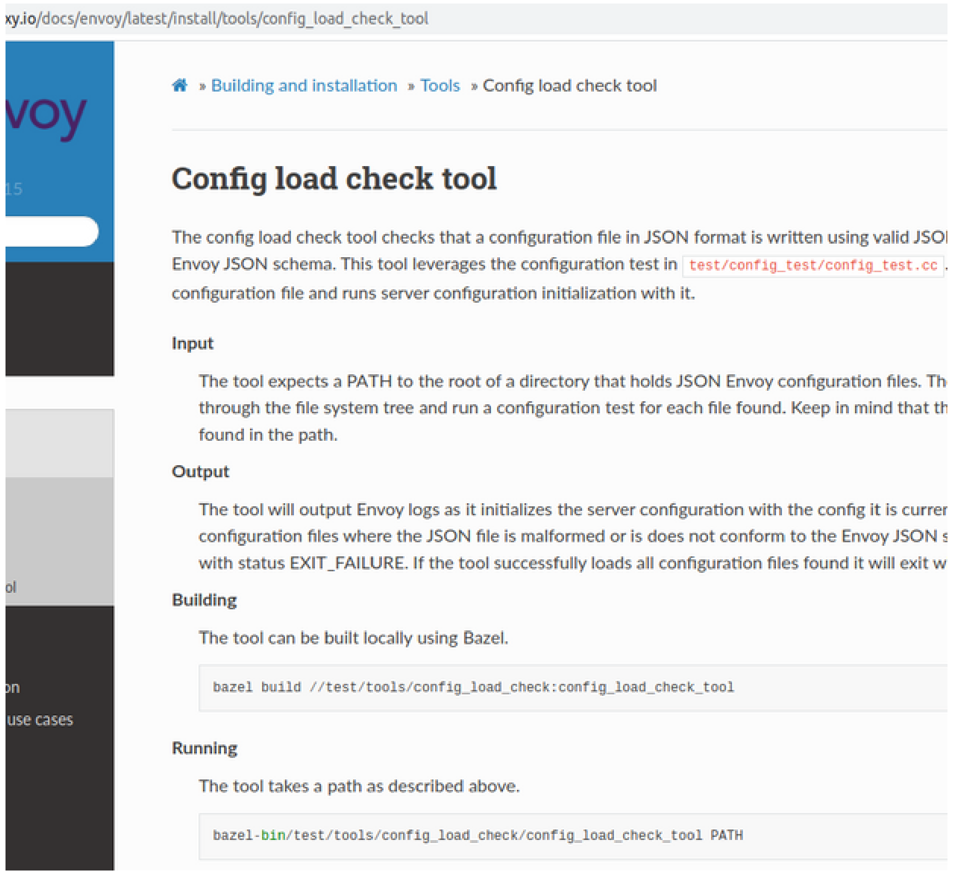
Make config deployments safe and self-service

Dead cluster/route check



- Checking for unused clusters can save you from tech debt and creating and sending unnecessary cluster configurations.
- Checking for unused routes can save you from 503ing traffic.
- Both checks can prevent human error from derailing a service launch or deprecation.

Validate bootstrap config



xy.io/docs/envoy/latest/install/tools/config_load_check_tool

voy

» Building and installation » Tools » Config load check tool

Config load check tool

The config load check tool checks that a configuration file in JSON format is written using valid JSOI
Envoy JSON schema. This tool leverages the configuration test in `test/config_test/config_test.cc` .
configuration file and runs server configuration initialization with it.

Input

The tool expects a PATH to the root of a directory that holds JSON Envoy configuration files. Th
through the file system tree and run a configuration test for each file found. Keep in mind that th
found in the path.

Output

The tool will output Envoy logs as it initializes the server configuration with the config it is curre
configuration files where the JSON file is malformed or is does not conform to the Envoy JSON s
with status EXIT_FAILURE. If the tool successfully loads all configuration files found it will exit w

Building

The tool can be built locally using Bazel.

```
bazel build //test/tools/config_load_check:config_load_check_tool
```

Running

The tool takes a path as described above.

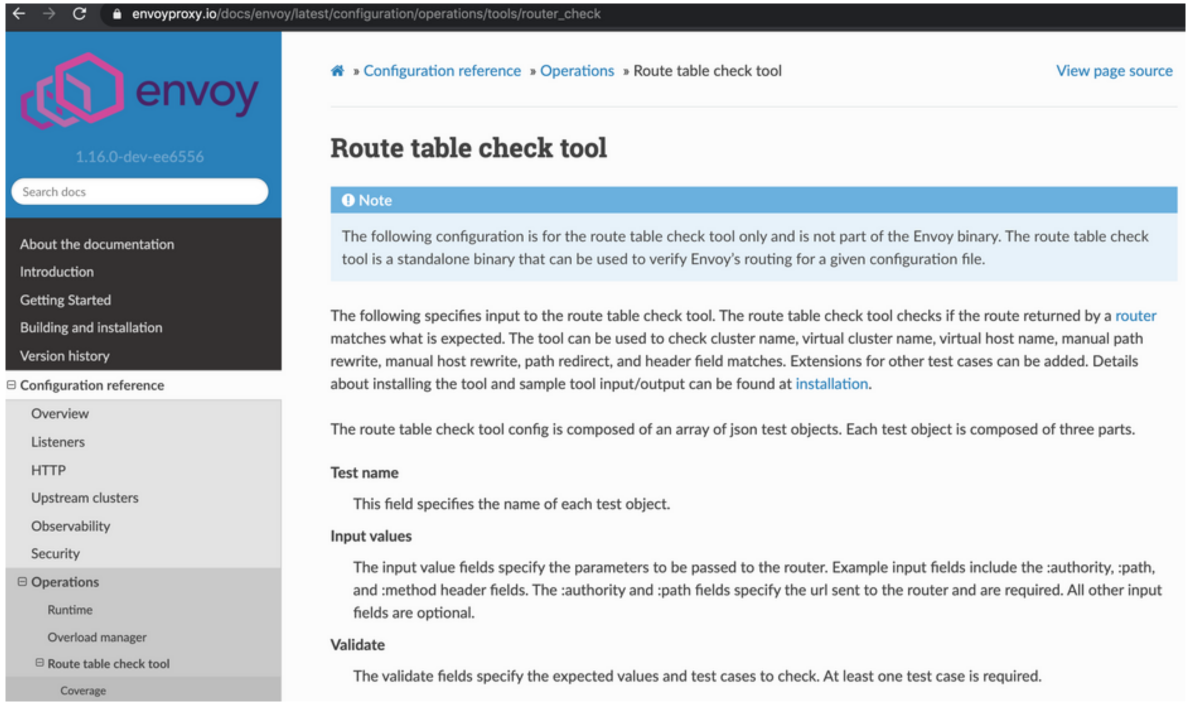
```
bazel-bin/test/tools/config_load_check/config_load_check_tool PATH
```

- **Envoy validation server**
  - runs envoy binary in config validation mode.
  - `--mode=validate` on bootstrap options
- **Test bootstrap config changes in PR**
  - /tmp/config_load_check_tool /code/envoy-static-config/generated/config

MADE WITH beautiful.ai

Router Check Tool

- Unit test routes
- Code coverage
- Deprecation check
- Test runtime-based routes



Route table check tool

Note

The following configuration is for the route table check tool only and is not part of the Envoy binary. The route table check tool is a standalone binary that can be used to verify Envoy's routing for a given configuration file.

The following specifies input to the route table check tool. The route table check tool checks if the route returned by a router matches what is expected. The tool can be used to check cluster name, virtual cluster name, virtual host name, manual path rewrite, manual host rewrite, path redirect, and header field matches. Extensions for other test cases can be added. Details about installing the tool and sample tool input/output can be found at installation.

The route table check tool config is composed of an array of json test objects. Each test object is composed of three parts.

Test name

This field specifies the name of each test object.

Input values

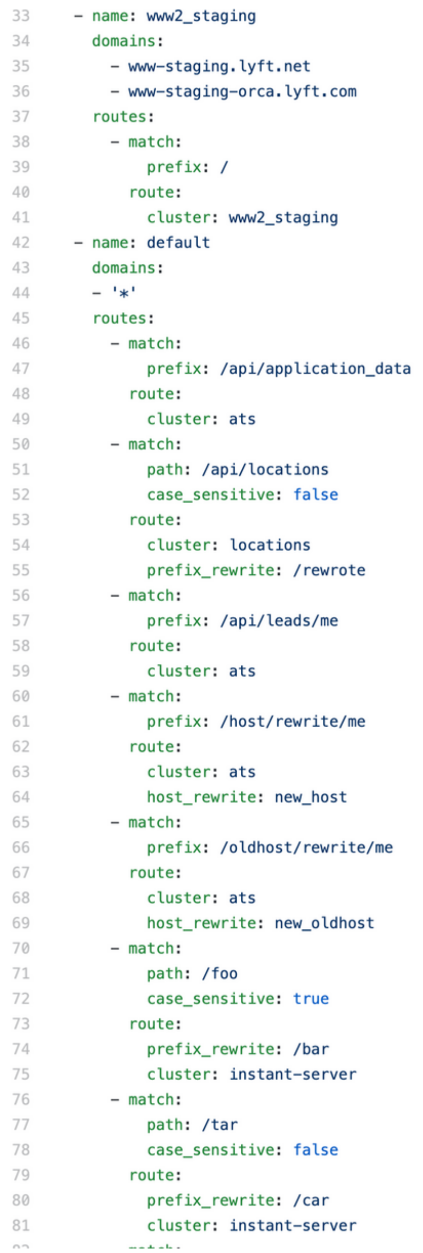
The input value fields specify the parameters to be passed to the router. Example input fields include the :authority, :path, and :method header fields. The :authority and :path fields specify the url sent to the router and are required. All other input fields are optional.

Validate

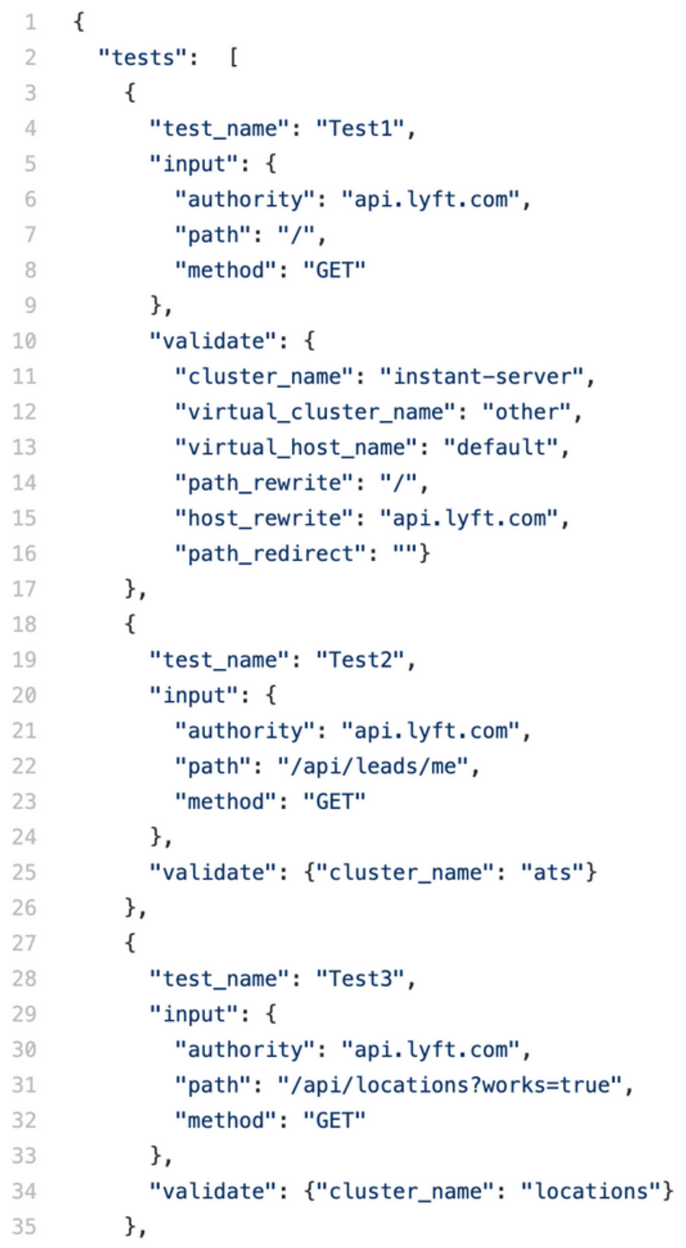
The validate fields specify the expected values and test cases to check. At least one test case is required.

Unit tested routes!!??

- ✓ prevent regressions
- ✓ self service route modifications
- ✓ iterate faster



```yaml
- name: www2_staging
  domains:
    - www-staging.lyft.net
    - www-staging-orca.lyft.com
  routes:
    - match:
        prefix: /
      route:
        cluster: www2_staging
- name: default
  domains:
  - '*'
  routes:
    - match:
        prefix: /api/application_data
      route:
        cluster: ats
    - match:
        path: /api/locations
        case_sensitive: false
      route:
        cluster: locations
        prefix_rewrite: /rewrote
    - match:
        prefix: /api/leads/me
      route:
        cluster: ats
    - match:
        prefix: /host/rewrite/me
      route:
        cluster: ats
        host_rewrite: new_host
    - match:
        prefix: /oldhost/rewrite/me
      route:
        cluster: ats
        host_rewrite: new_oldhost
    - match:
        path: /foo
        case_sensitive: true
      route:
        prefix_rewrite: /bar
        cluster: instant-server
    - match:
        path: /tar
        case_sensitive: false
      route:
        prefix_rewrite: /car
        cluster: instant-server
```



```json
{
  "tests":  [
    {
      "test_name": "Test1",
      "input": {
        "authority": "api.lyft.com",
        "path": "/",
        "method": "GET"
      },
      "validate": {
        "cluster_name": "instant-server",
        "virtual_cluster_name": "other",
        "virtual_host_name": "default",
        "path_rewrite": "/",
        "host_rewrite": "api.lyft.com",
        "path_redirect": ""}
    },
    {
      "test_name": "Test2",
      "input": {
        "authority": "api.lyft.com",
        "path": "/api/leads/me",
        "method": "GET"
      },
      "validate": {"cluster_name": "ats"}
    },
    {
      "test_name": "Test3",
      "input": {
        "authority": "api.lyft.com",
        "path": "/api/locations?works=true",
        "method": "GET"
      },
      "validate": {"cluster_name": "locations"}
    },
```

MADE WITH beautiful.ai

Now run the tests!

```
router_check/router_check_tool -c config/TestRoutes.yaml -t
config/TestRoutes.golden.proto.json --details

Test1

Test2

Current route coverage: 64.7059%

~ echo $?

~ 0
```

```
router_check_tool -c config/TestRoutes.yaml -t config/TestRoutes.golden.proto.json
--only-show-failures

Test1

expected: [instant-server-fail], actual: [instant-server], test type: cluster_name

~ echo $?

~ 1
```

MADE WITH beautiful.ai

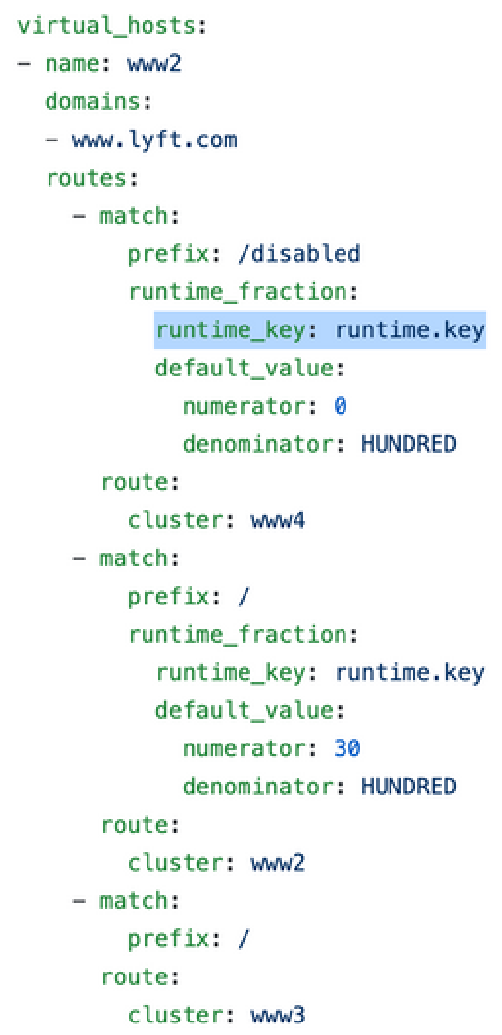
Code coverage!!

```
router_check/router_check_tool -c config/TestRoutes.yaml -t config/TestRoutes.golden.proto.json --details

Test1

Test2

Current route coverage: 64.7059%

~ echo $?

~ 0
```

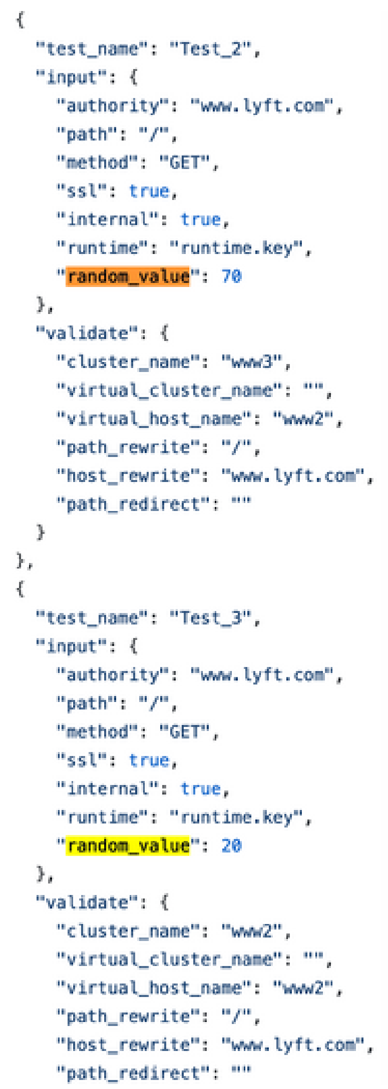
```
router_check/router_check_tool -c config/TestRoutes.yaml -t config/TestRoutes.golden.proto.json --fail-under 80

Current route coverage: 64.7059%

Failed to meet coverage requirement: 80%

~ echo $?

~ 1
```

```
router_check/router_check_tool -c config/TestRoutes.yaml -t config/TestRoutes.golden.proto.json --covall

Missing test for host: www2_staging, route: prefix: "/"

Missing test for host: default, route: path: "/tar"case_sensitive {}

Current route coverage: 25.8824%

~ echo $1

~ 0
```

Runtime-based routes!!

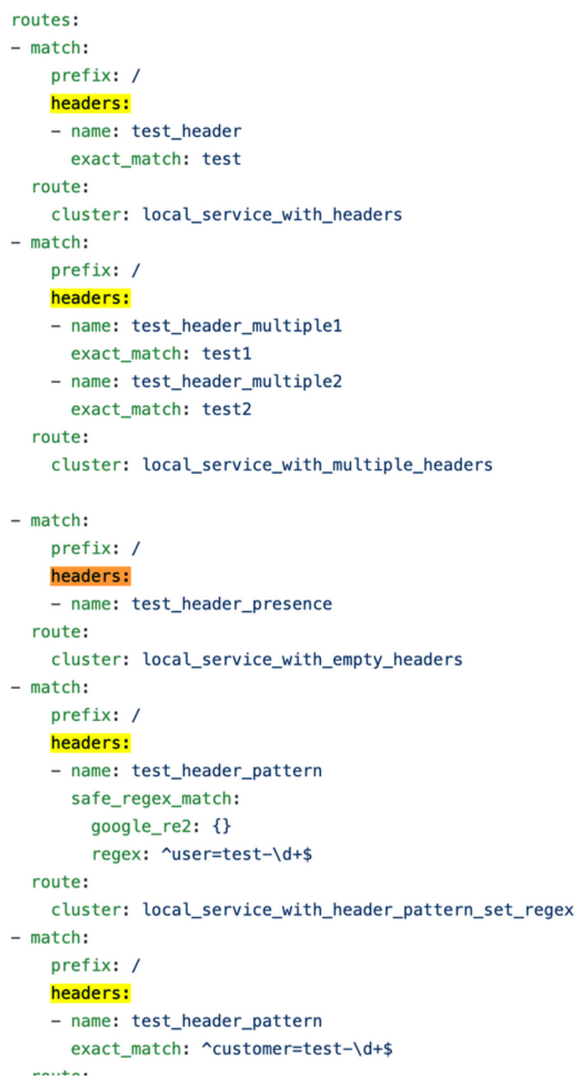


```yaml
virtual_hosts:
- name: www2
  domains:
  - www.lyft.com
  routes:
    - match:
        prefix: /disabled
        runtime_fraction:
          runtime_key: runtime.key
          default_value:
            numerator: 0
            denominator: HUNDRED
      route:
        cluster: www4
    - match:
        prefix: /
        runtime_fraction:
          runtime_key: runtime.key
          default_value:
            numerator: 30
            denominator: HUNDRED
      route:
        cluster: www2
    - match:
        prefix: /
      route:
        cluster: www3
```

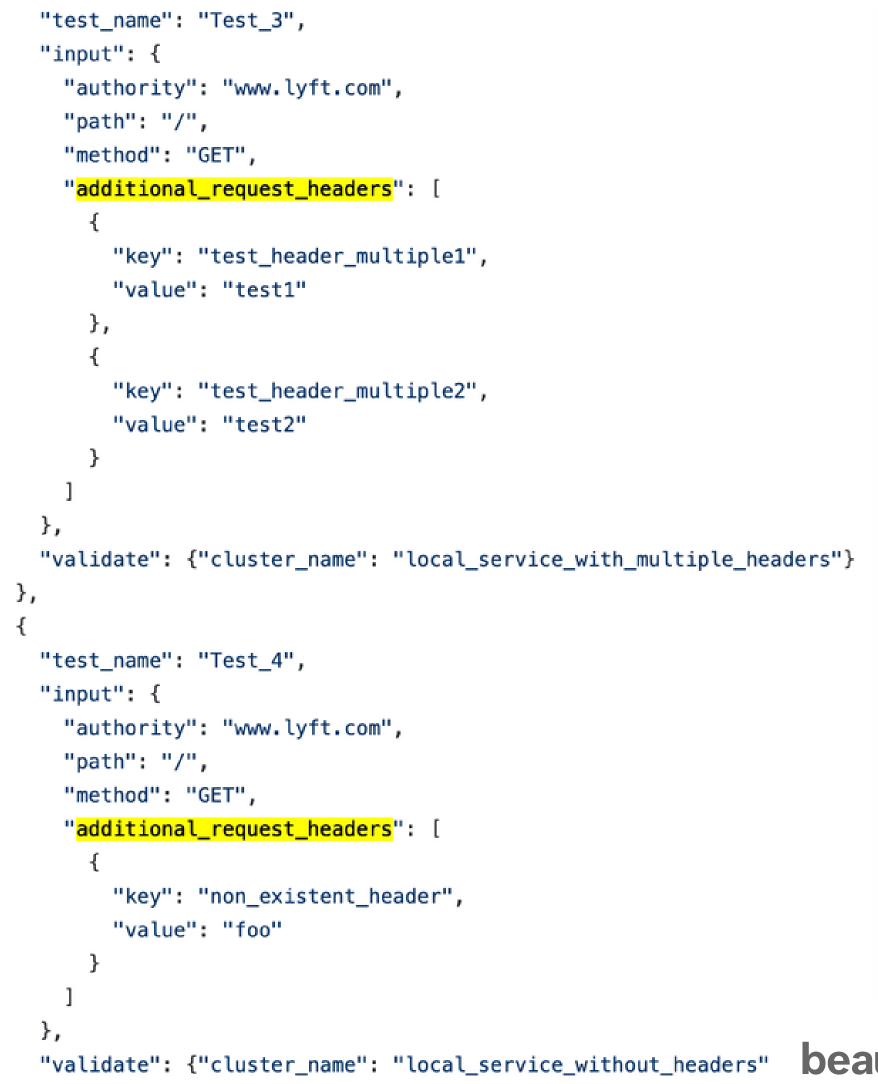


```json
{
  "test_name": "Test_2",
  "input": {
    "authority": "www.lyft.com",
    "path": "/",
    "method": "GET",
    "ssl": true,
    "internal": true,
    "runtime": "runtime.key",
    "random_value": 70
  },
  "validate": {
    "cluster_name": "www3",
    "virtual_cluster_name": "",
    "virtual_host_name": "www2",
    "path_rewrite": "/",
    "host_rewrite": "www.lyft.com",
    "path_redirect": ""
  }
},
{
  "test_name": "Test_3",
  "input": {
    "authority": "www.lyft.com",
    "path": "/",
    "method": "GET",
    "ssl": true,
    "internal": true,
    "runtime": "runtime.key",
    "random_value": 20
  },
  "validate": {
    "cluster_name": "www2",
    "virtual_cluster_name": "",
    "virtual_host_name": "www2",
    "path_rewrite": "/",
    "host_rewrite": "www.lyft.com",
    "path_redirect": ""
```

Header tests!!



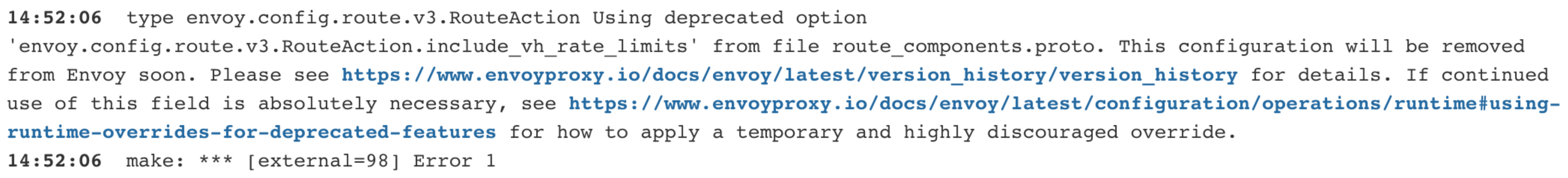
```yaml
routes:
- match:
    prefix: /
    headers:
    - name: test_header
      exact_match: test
  route:
    cluster: local_service_with_headers
- match:
    prefix: /
    headers:
    - name: test_header_multiple1
      exact_match: test1
    - name: test_header_multiple2
      exact_match: test2
  route:
    cluster: local_service_with_multiple_headers

- match:
    prefix: /
    headers:
    - name: test_header_presence
  route:
    cluster: local_service_with_empty_headers
- match:
    prefix: /
    headers:
    - name: test_header_pattern
      safe_regex_match:
        google_re2: {}
        regex: ^user=test-\d+$
  route:
    cluster: local_service_with_header_pattern_set_regex
- match:
    prefix: /
    headers:
    - name: test_header_pattern
      exact_match: ^customer=test-\d+$
```



```json
    "test_name": "Test_3",
    "input": {
      "authority": "www.lyft.com",
      "path": "/",
      "method": "GET",
      "additional_request_headers": [
        {
          "key": "test_header_multiple1",
          "value": "test1"
        },
        {
          "key": "test_header_multiple2",
          "value": "test2"
        }
      ]
    },
    "validate": {"cluster_name": "local_service_with_multiple_headers"}
  },
  {
    "test_name": "Test_4",
    "input": {
      "authority": "www.lyft.com",
      "path": "/",
      "method": "GET",
      "additional_request_headers": [
        {
          "key": "non_existent_header",
          "value": "foo"
        }
      ]
    },
    "validate": {"cluster_name": "local_service_without_headers"
```

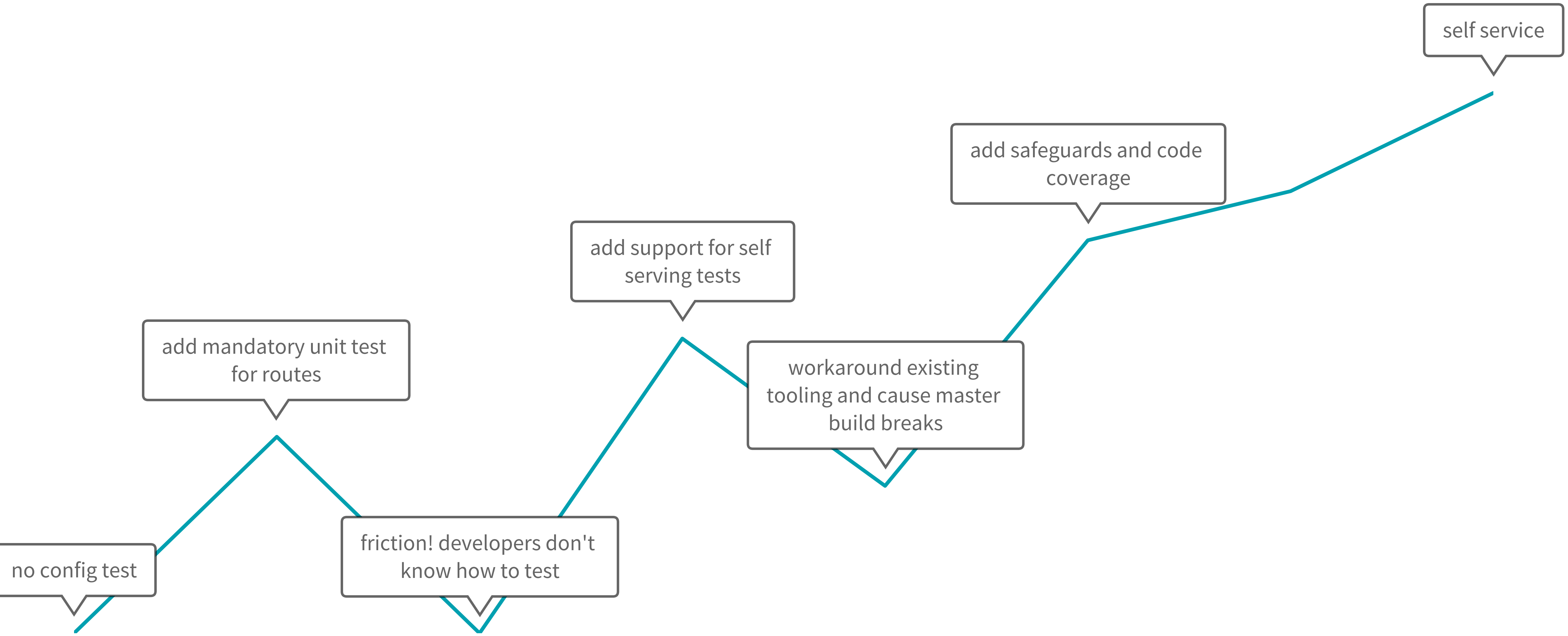
Field deprecation check

- keep up-to-date with Envoy's field deprecation cycle
- turned on by default. Turn it off by using *--disable-deprecation-check*
- reduces tech debt



```
14:52:06  type envoy.config.route.v3.RouteAction Using deprecated option
'envoy.config.route.v3.RouteAction.include_vh_rate_limits' from file route_components.proto. This configuration will be removed
from Envoy soon. Please see https://www.envoyproxy.io/docs/envoy/latest/version_history/version_history for details. If continued
use of this field is absolutely necessary, see https://www.envoyproxy.io/docs/envoy/latest/configuration/operations/runtime#using-
runtime-overrides-for-deprecated-features for how to apply a temporary and highly discouraged override.
14:52:06  make: *** [external=98] Error 1
```

Culture Shift


no config test
add mandatory unit test for routes
friction! developers don't know how to test
add support for self serving tests
workaround existing tooling and cause master build breaks
add safeguards and code coverage
self service

Future Direction

Test support for one-off features

- **CORS, route types, route properties**
  Increase what behavior can be tested.

Utilize production code

- **Keep up with changes in production behavior so that testing behavior does not diverge.**
  Testing support is thus consistent with the code that actually runs in prod, e.g. routing code or header-altering code. Test support for various features becomes less brittle.

True blackbox testing

- **Input a full Envoy config versus specific route inputs known beforehand.**
  Users can simulate request behavior without knowing the specific test cases beforehand and inspect the resultant response. This approach also lends itself well to a UI-based way of testing the routing table.

Questions?