Continuous verification for serverless applications

Serverless WEEK

Gunnar Grosch
@gunnargrosch
October 30, 2020

“Testing to ensure that you can meet your availability goals is the only way you can have confidence that you will meet those goals”

Reliability Pillar
AWS Well-Architected Framework

Verification of applications

Verification of applications

Verification of applications

Conditions for verification

Errors in your code

Security policy errors

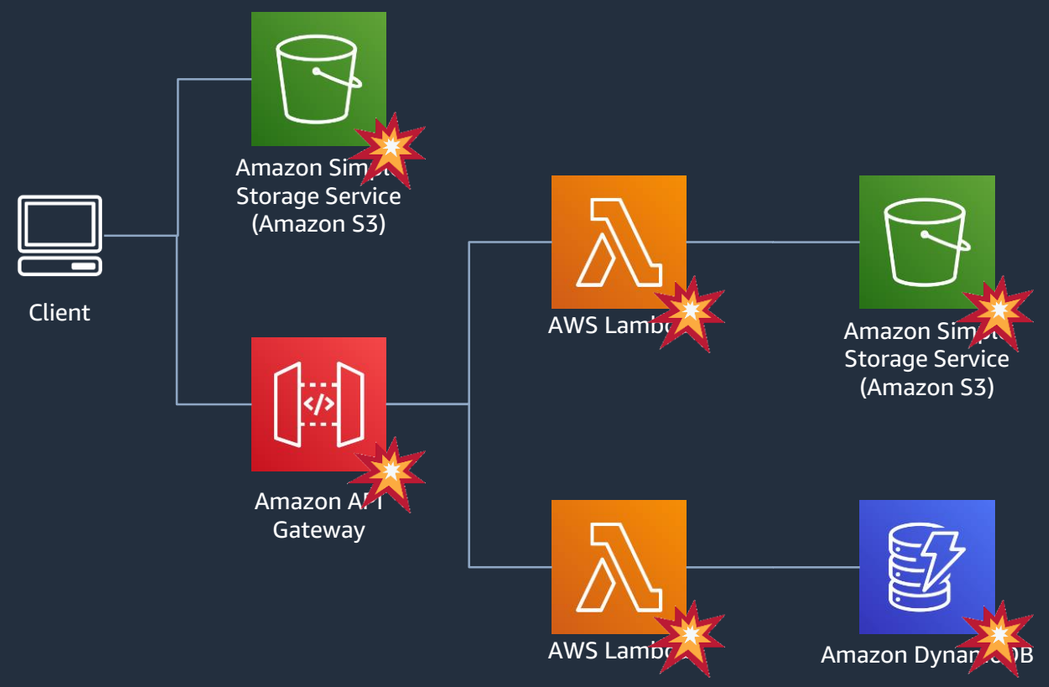
Service configuration errors

Function disk space failure

Downstream service issues

Concurrency and throttling

Latency



Tools for adding conditions

Chaos-lambda	Failure-lambda
Python	NodeJS

Failure-lambda NodeJS

NPM package for NodeJS Lambdas

Configuration using Parameter Store or AWS AppConfig

Several failure modes

- Latency
- Status code
- Exception
- Disk space
- Denylist

```
const failureLambda = require('failure-lambda’)
exports.handler = failureLambda(async (event, context) => {
  ...
})
```

```
{
  "isEnabled": false,
  "failureMode": "latency",
  "rate": 1,
  "minLatency": 100,
  "maxLatency": 400,
  "exceptionMsg": "Exception message!",
  "statusCode": 404,
  "diskSpace": 100,
  “denylist": [
    "s3.*.amazonaws.com",
    "dynamodb.*.amazonaws.com"
  ]
}
```

Verification of applications

Verification example

Objective: My purchase API should respond in less than 400 ms

Measure: 400 ms threshold

Condition: 100-200 ms latency injection to function

Verify: Pass or fail

Verification example

Objective: I want 10 clicks per second

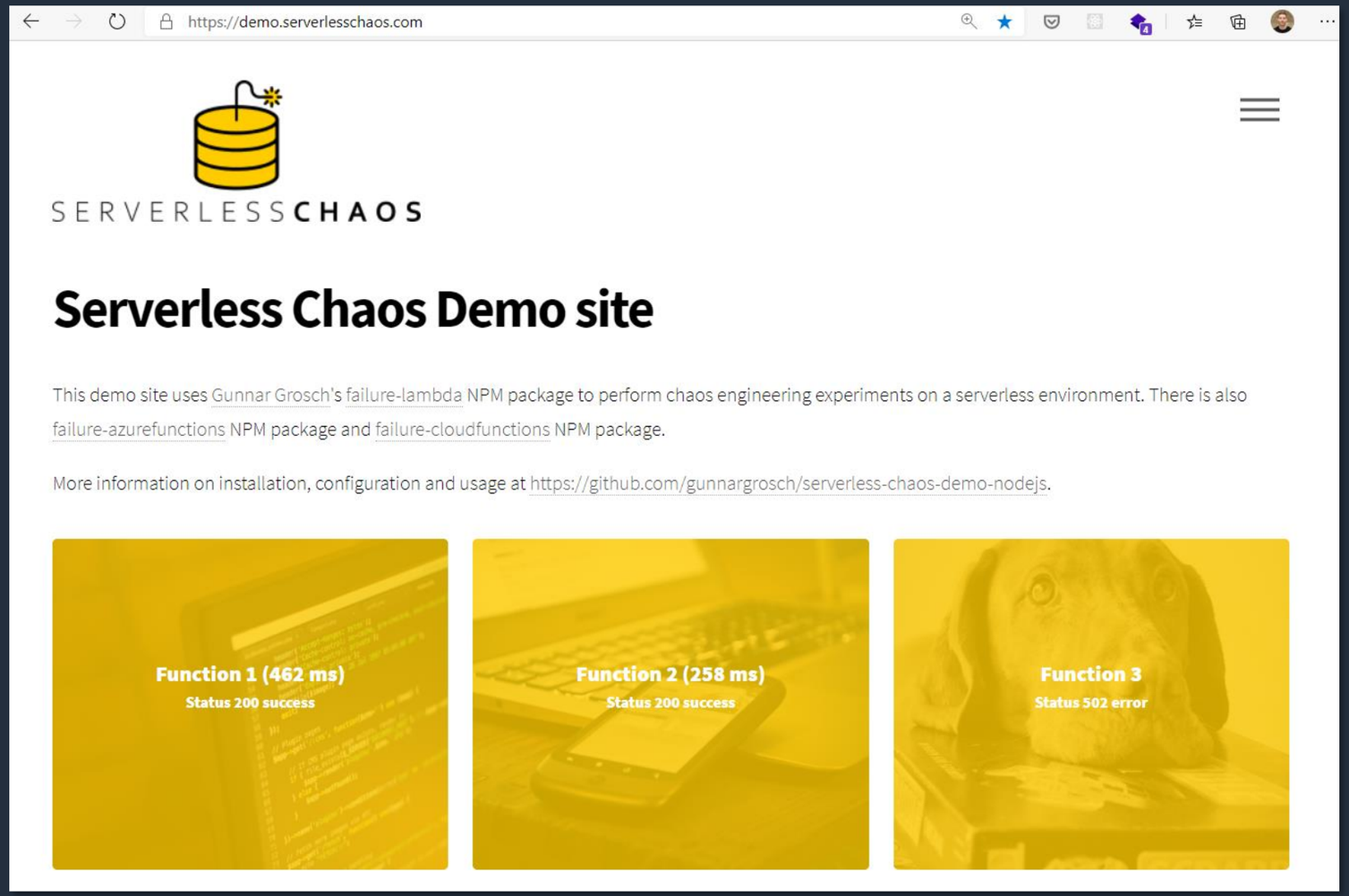
Measure: 10 clicks threshold

Condition: Inject downstream service failure

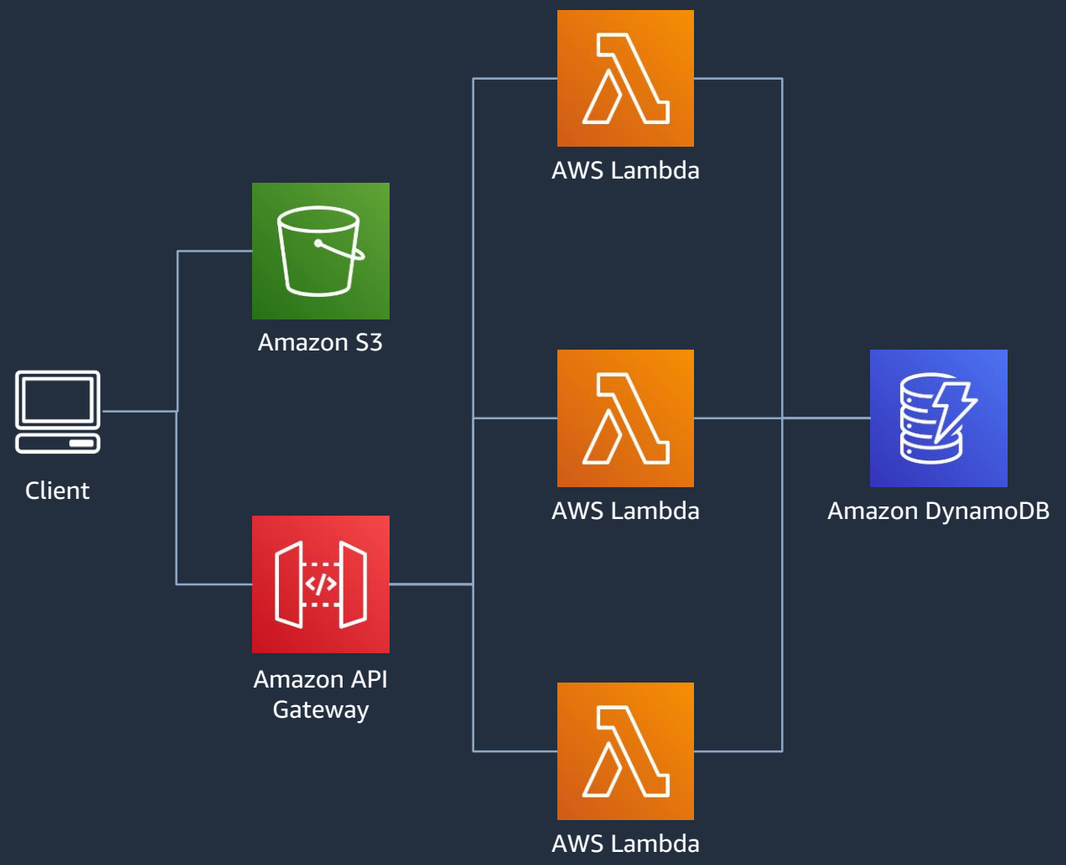
Verify: Pass or fail

Verification demo

Verification demo

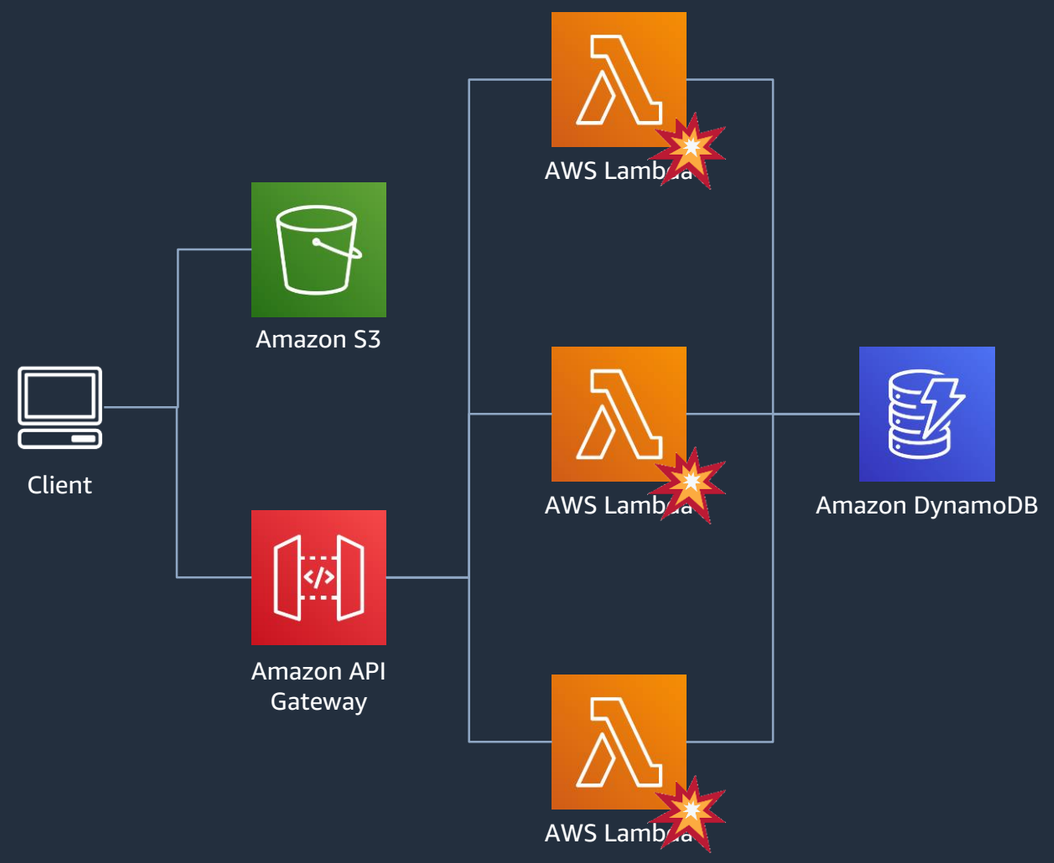


Verification demo



Verification demo

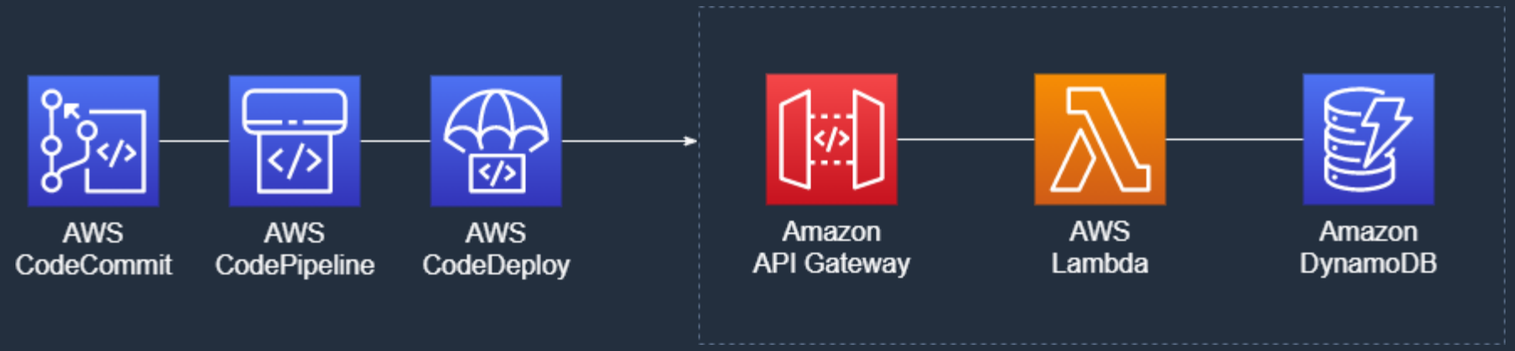
- Condition: Add 100-400 ms latency for each invocation
- Condition: Return error codes on some invocations
- Condition: Intercept and deny connections to DynamoDB



Demo

Making verification continuous

Making verification continuous



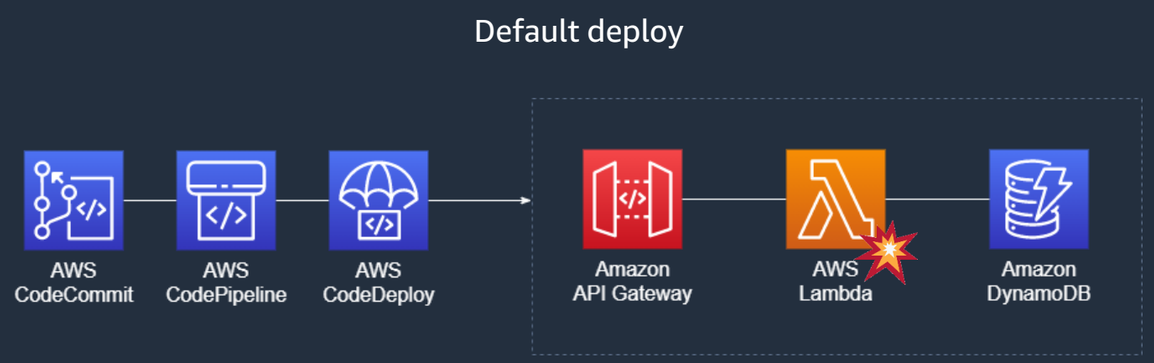
Making verification continuous

Objective: My purchase API should respond in less than 400 ms

Measure: 400 ms threshold

Condition: 100-200 ms latency injection to function

Verify: Pass



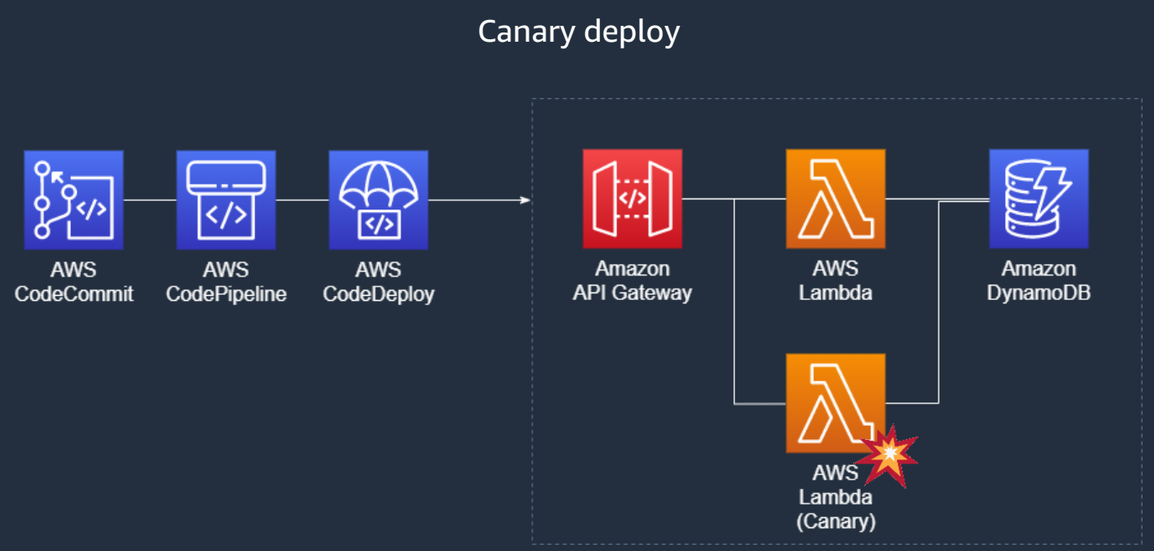
Making verification continuous

Objective: My purchase API should respond in less than 400 ms

Measure: 400 ms threshold

Condition: 100-200 ms latency injection to function

Verify: Fail



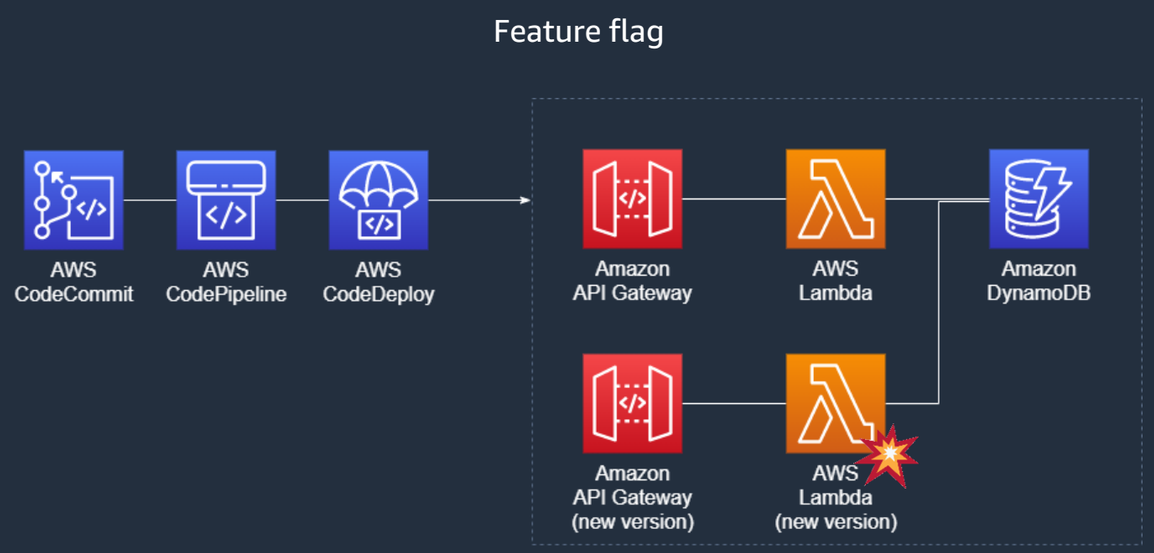
Making verification continuous

Objective: My purchase API should respond in less than 400 ms

Measure: 400 ms threshold

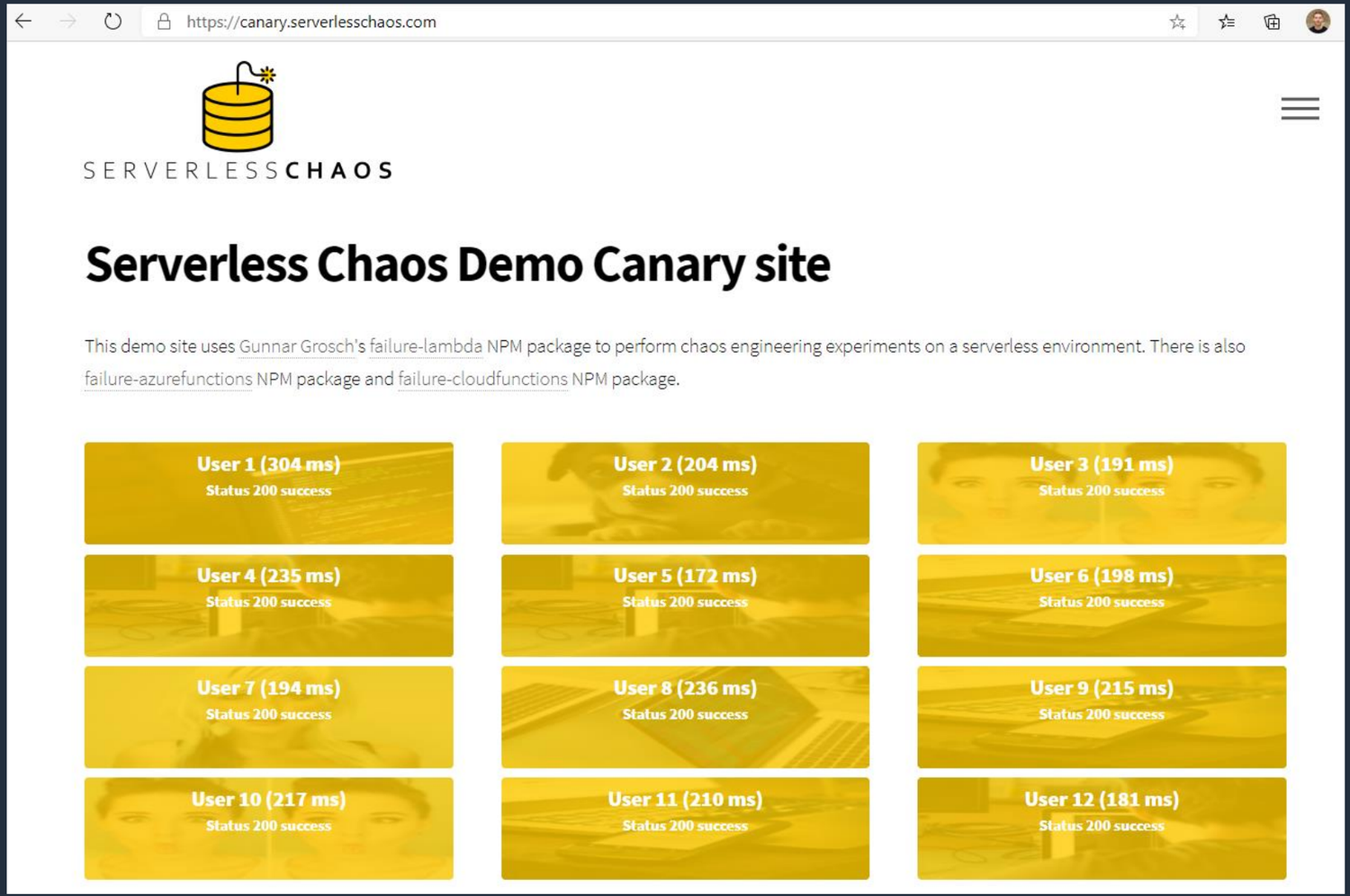
Condition: 100-200 ms latency injection to function

Verify: Pass

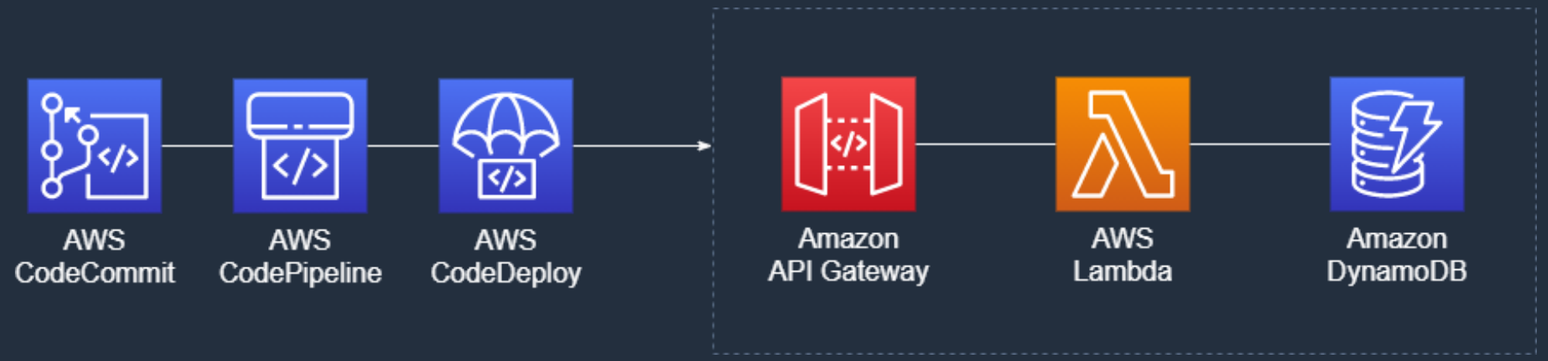


Continuous verification demo

Continuous verification demo



Continuous verification demo



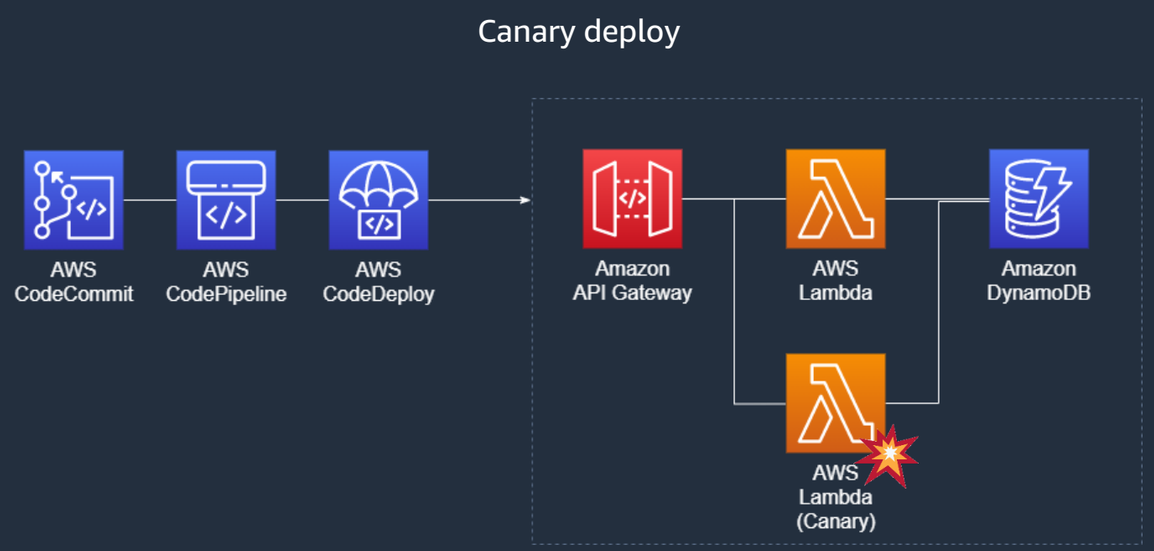
Continuous verification demo

Objective: Users should frequently get new images

Measure: 10 images loaded per minute

Condition: Return error codes on invocations

Verify: Pass or Fail



Demo

Key learnings

Verification tells you why it’s important to explore a set of conditions

It’s not about breaking things, it’s about learning and building confidence

It’s easy to get started.

“Testing to ensure that you can meet your availability goals is the only way you can have confidence that you will meet those goals”

Reliability Pillar
AWS Well-Architected Framework

Do you want more?

Reliability pillar
https://docs.aws.amazon.com/wellarchitected/latest/reliability-pillar/welcome.html

Serverless Chaos Demo app
https://demo.serverlesschaos.com

Failure-lambda
https://github.com/gunnargrosch/failure-lambda

Chaos-lambda
https://github.com/adhorn/aws-lambda-chaos-injection/

Thank you!

Gunnar Grosch
@gunnargrosch