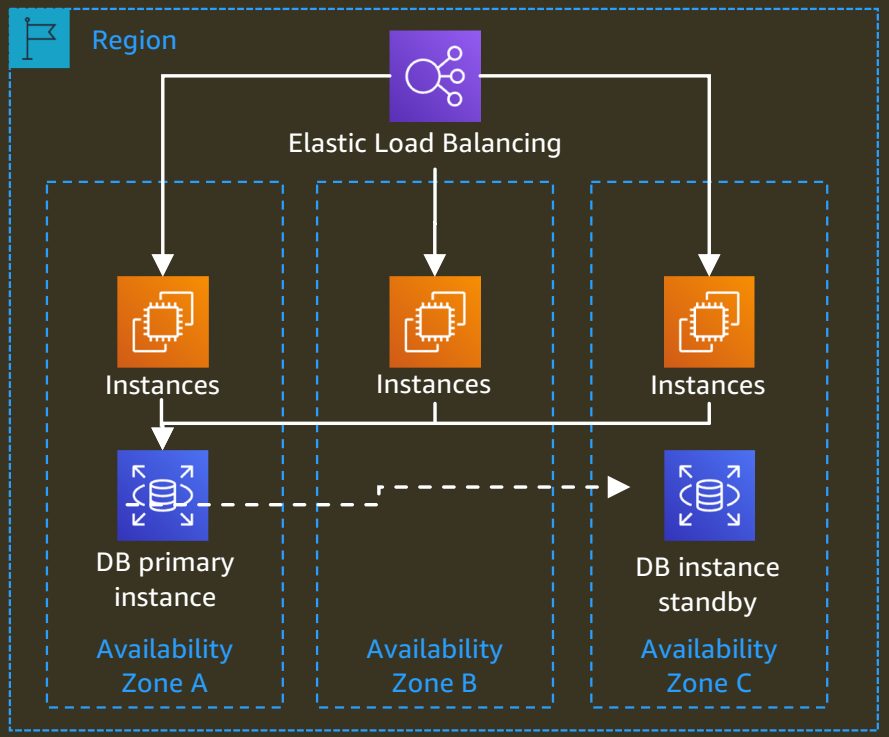
How to Test Your Fault Isolation Boundaries in the Cloud

jason_barto

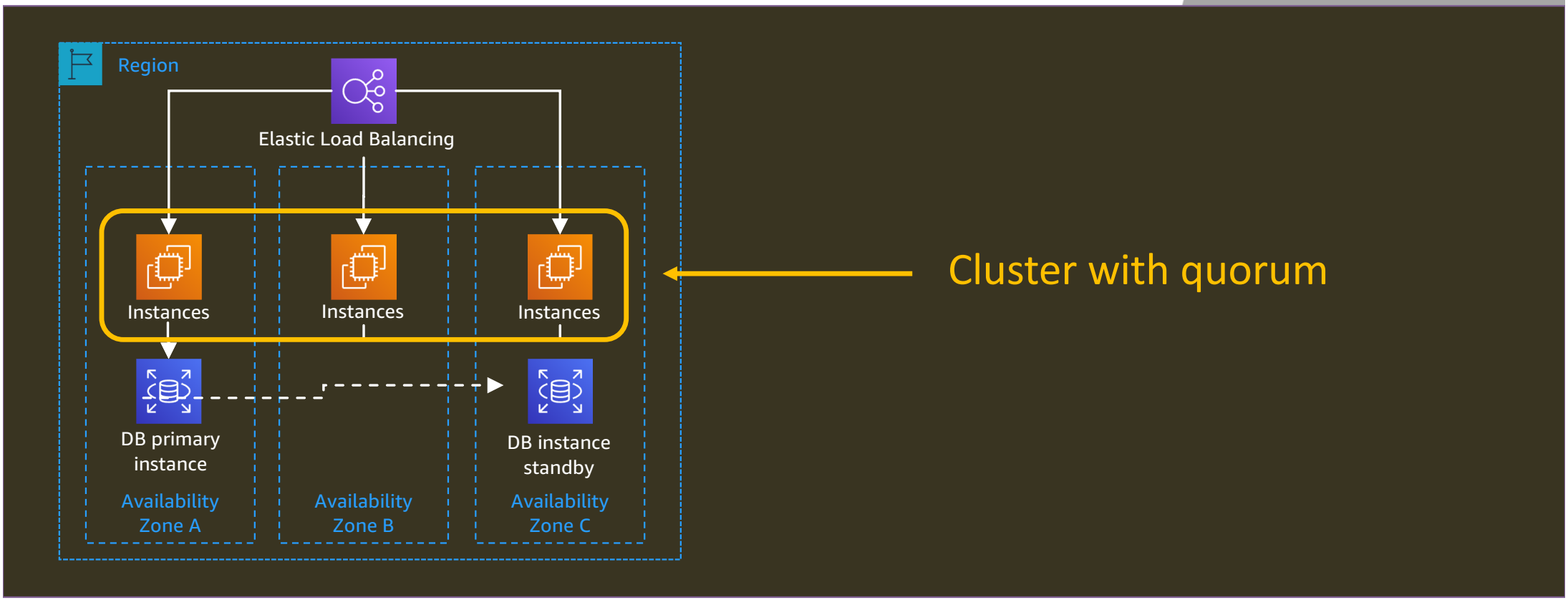
A Tale of Two Systems

How a resilient banking system failed

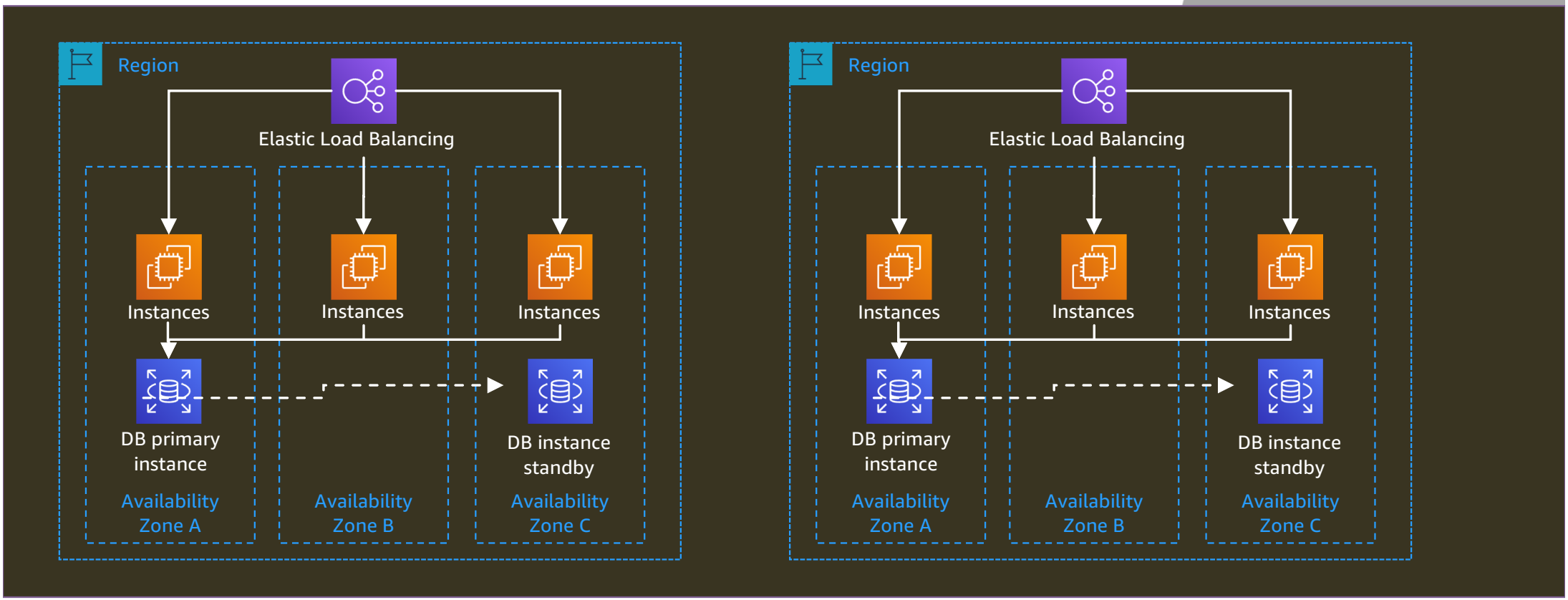
A Tale of Two Systems



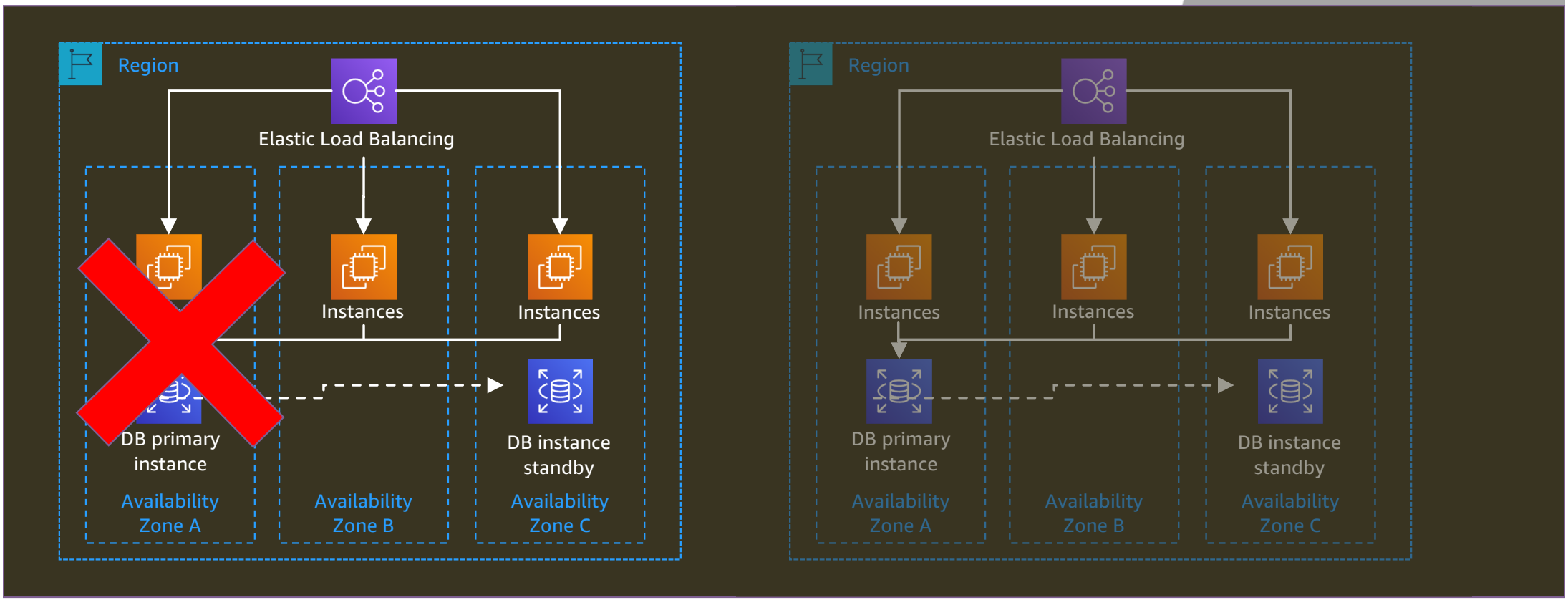
A Tale of Two Systems



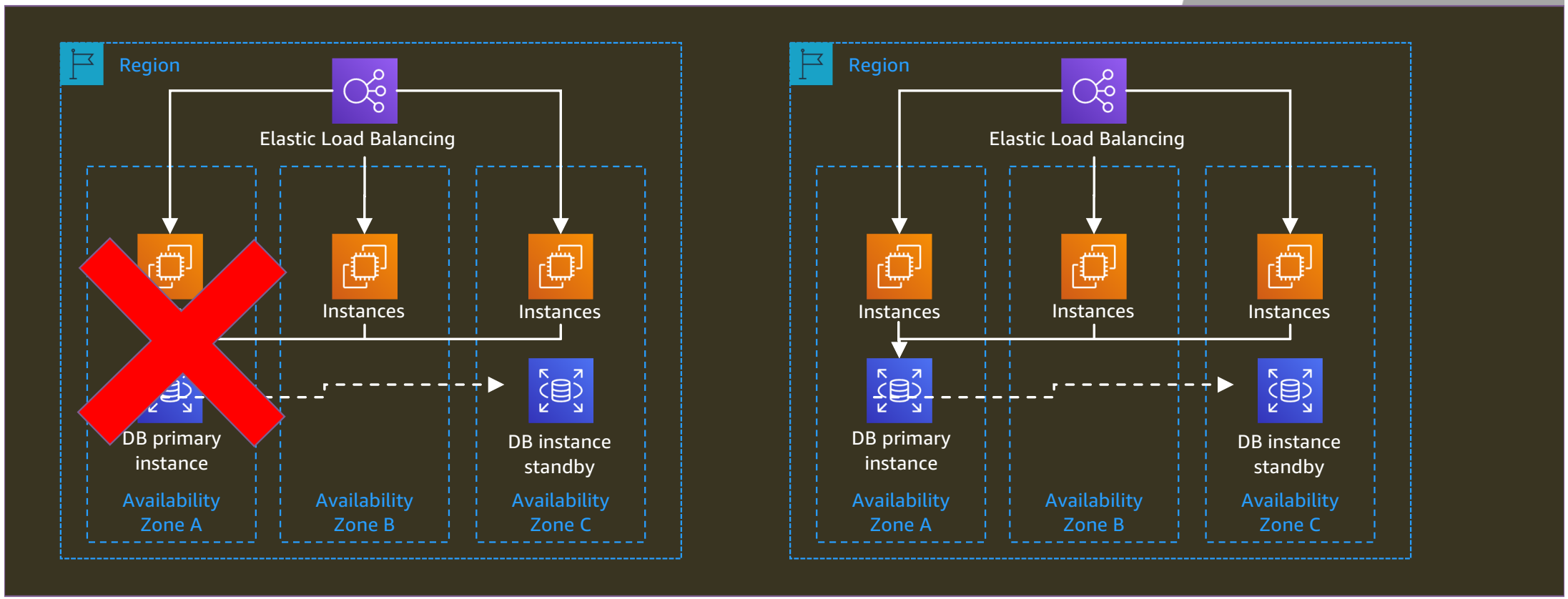
A Tale of Two Systems



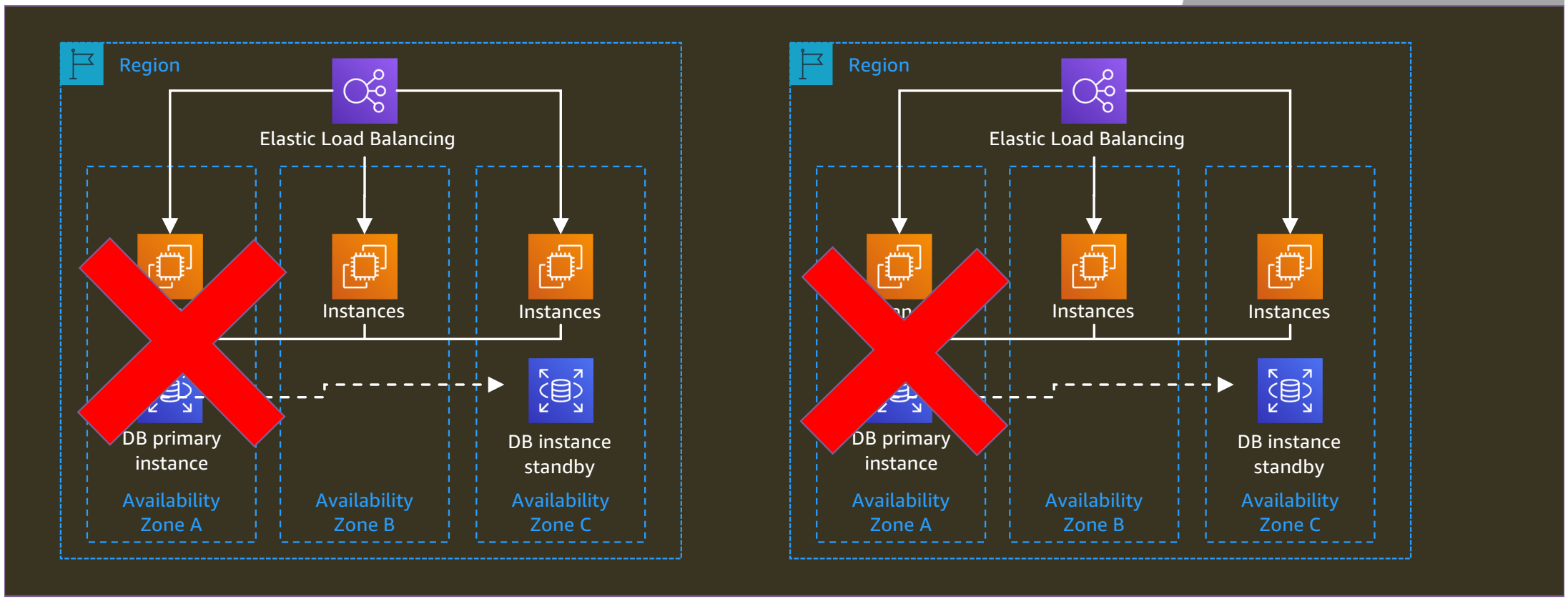
A Tale of Two Systems

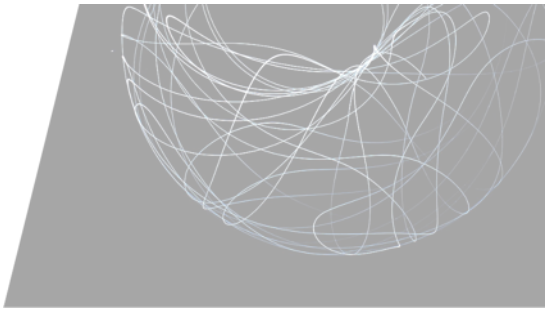


A Tale of Two Systems

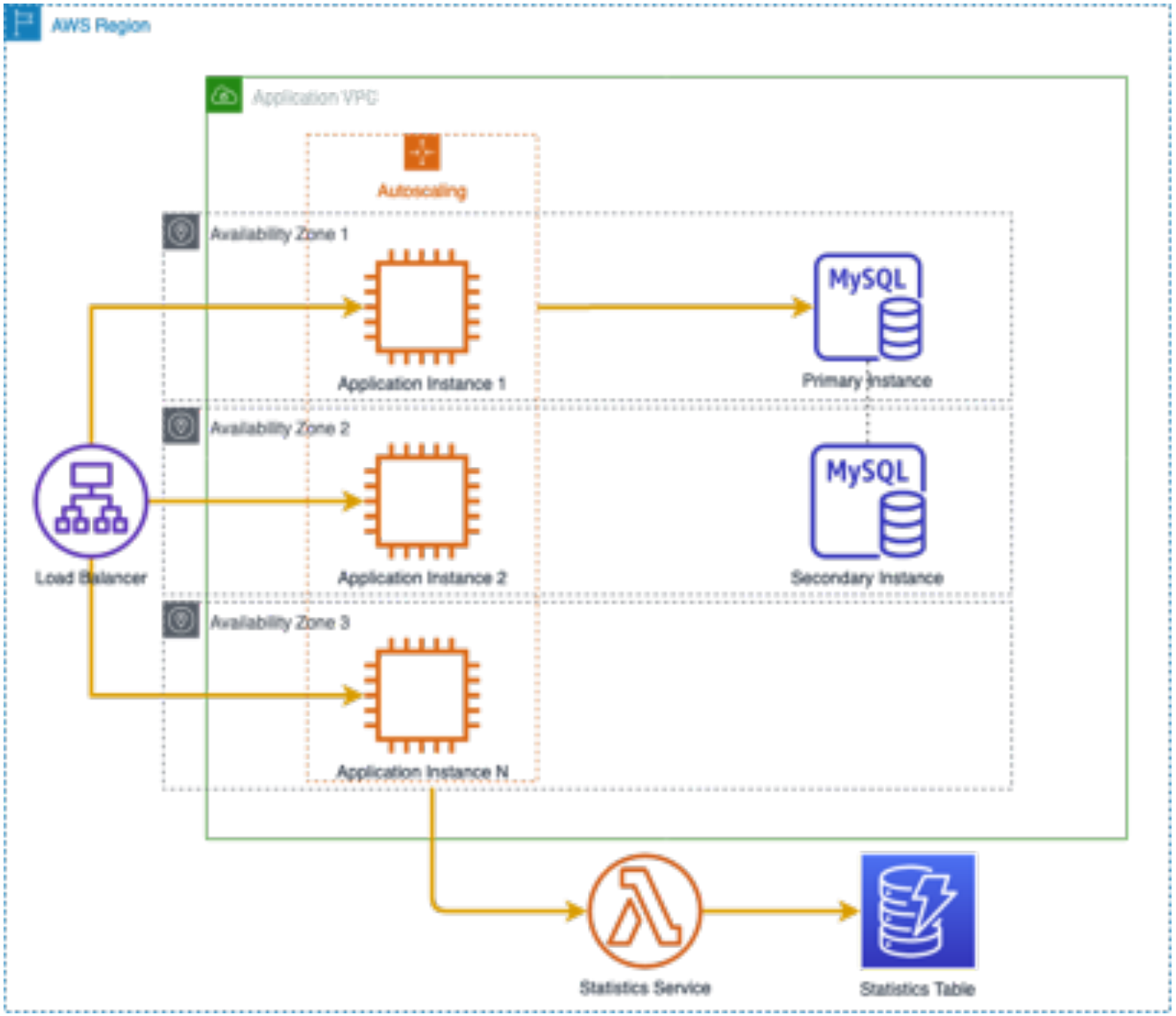


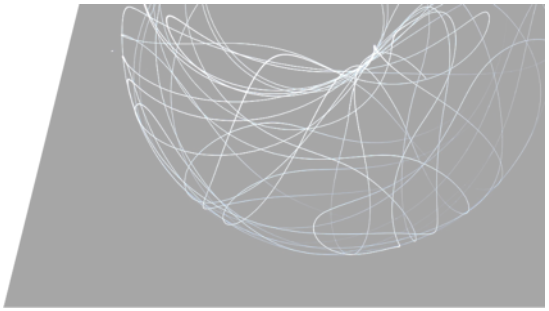
A Tale of Two Systems





So how do we confidently create resilient systems?

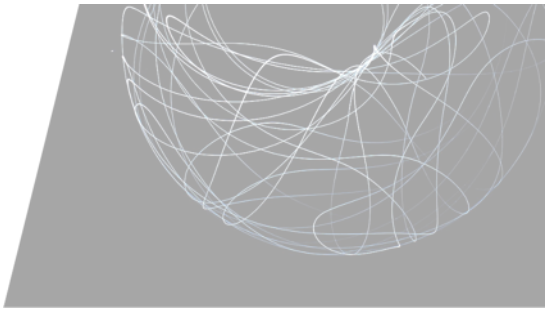




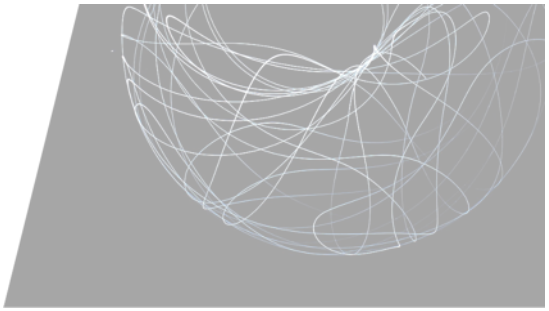
1. Failure domains
2. Fault isolation boundaries
3. Testing with chaos experiments

Failure Domains

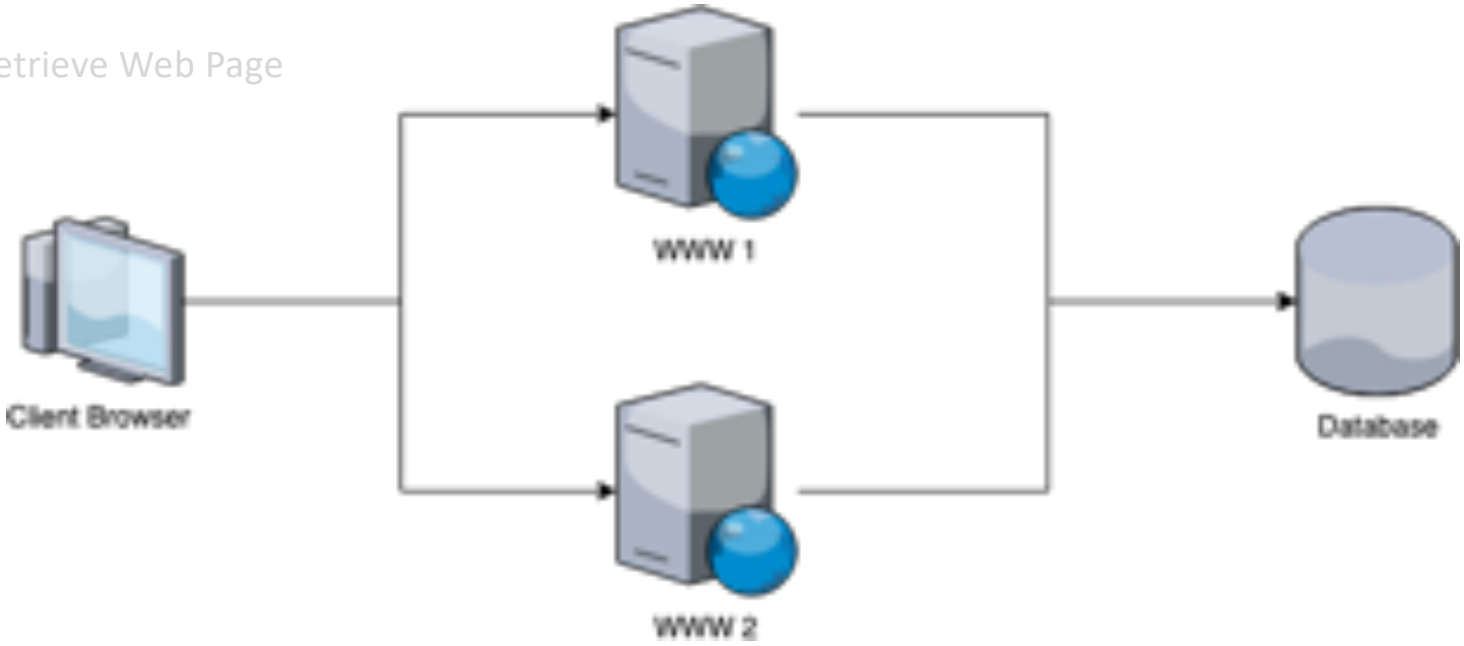
Failure Domain



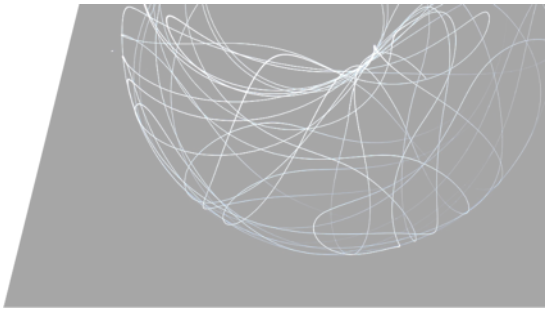
Failure Domain



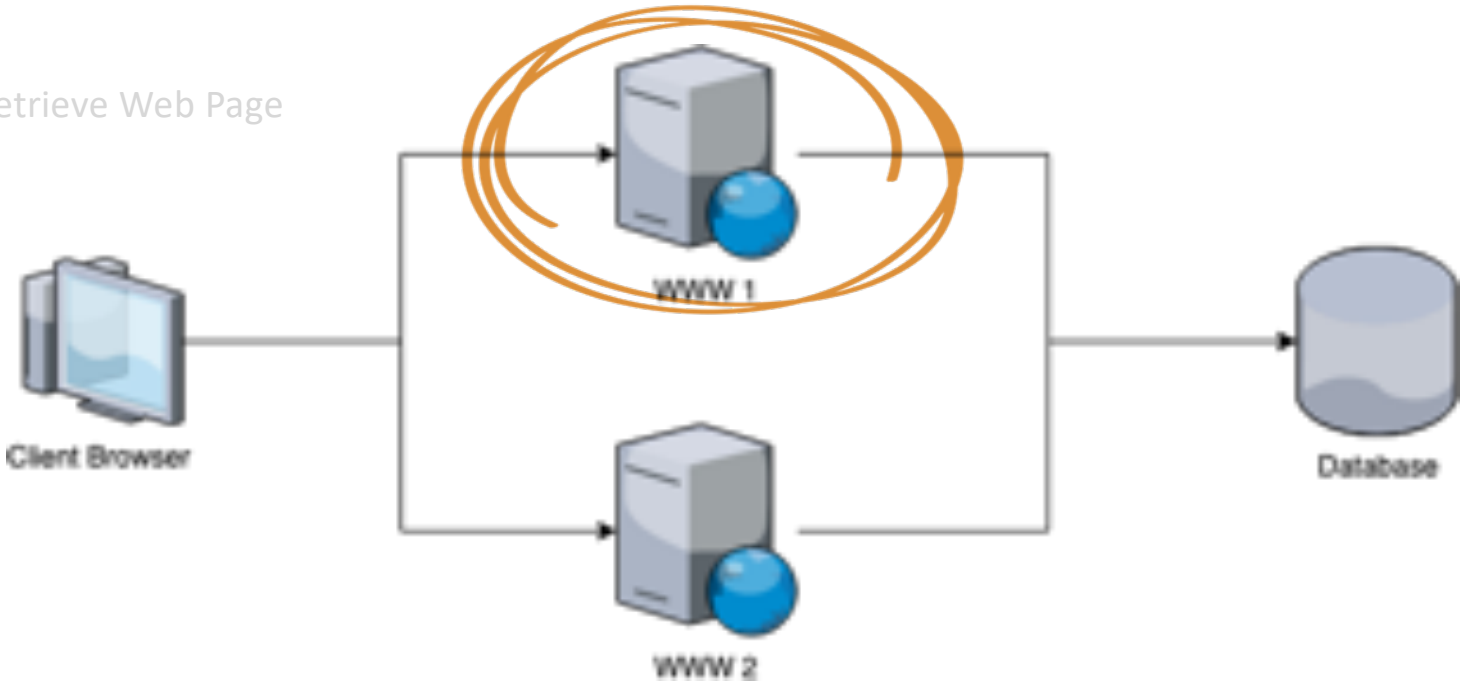
Scenario: Retrieve Web Page



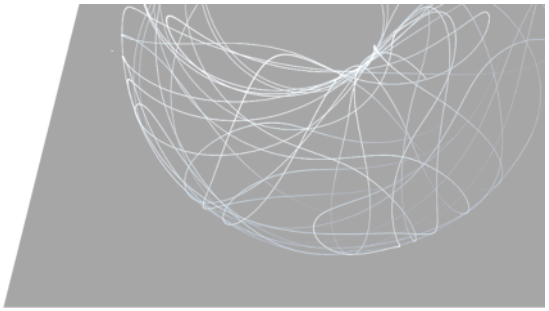
Failure Domain



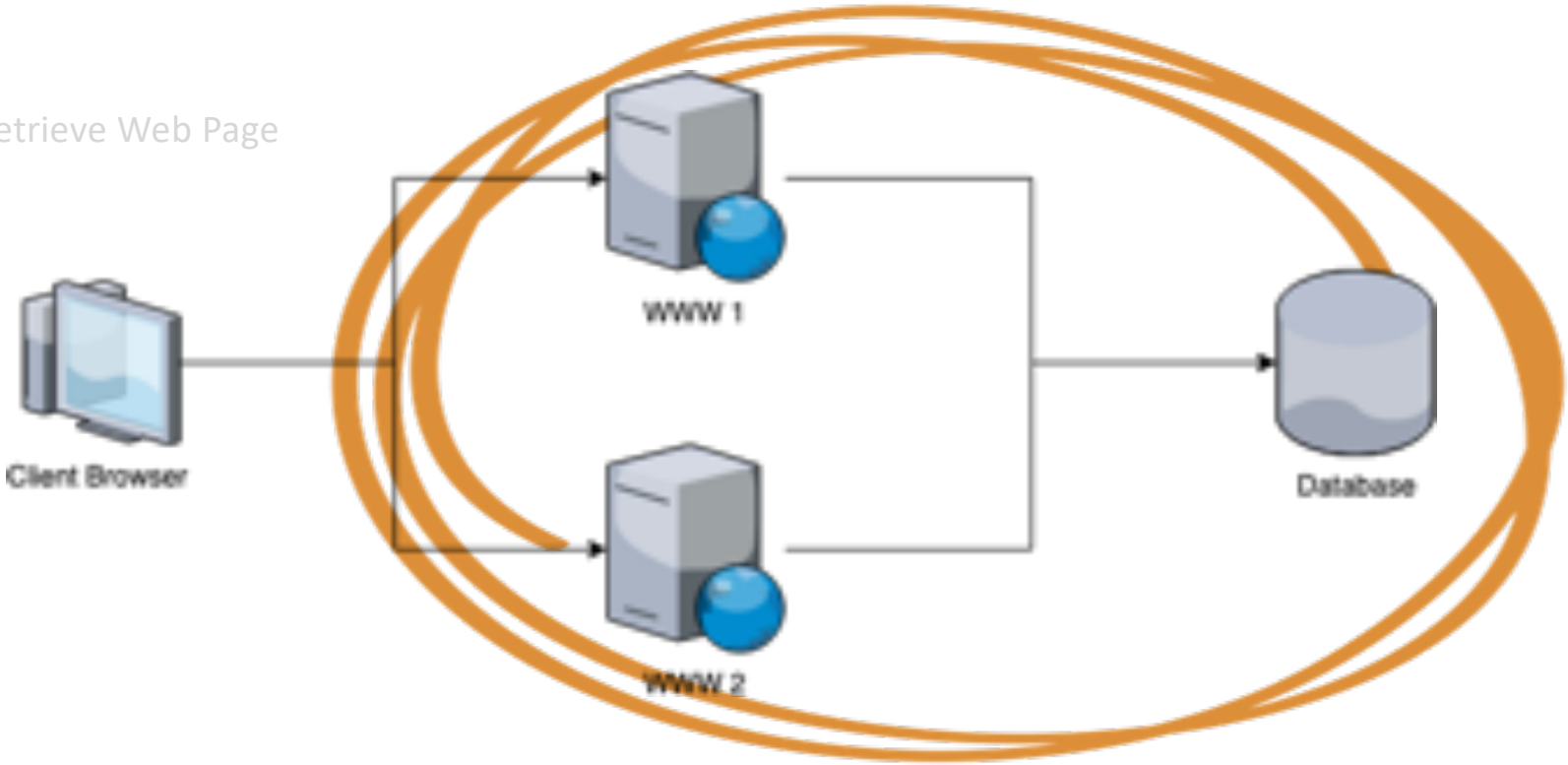
Scenario: Retrieve Web Page

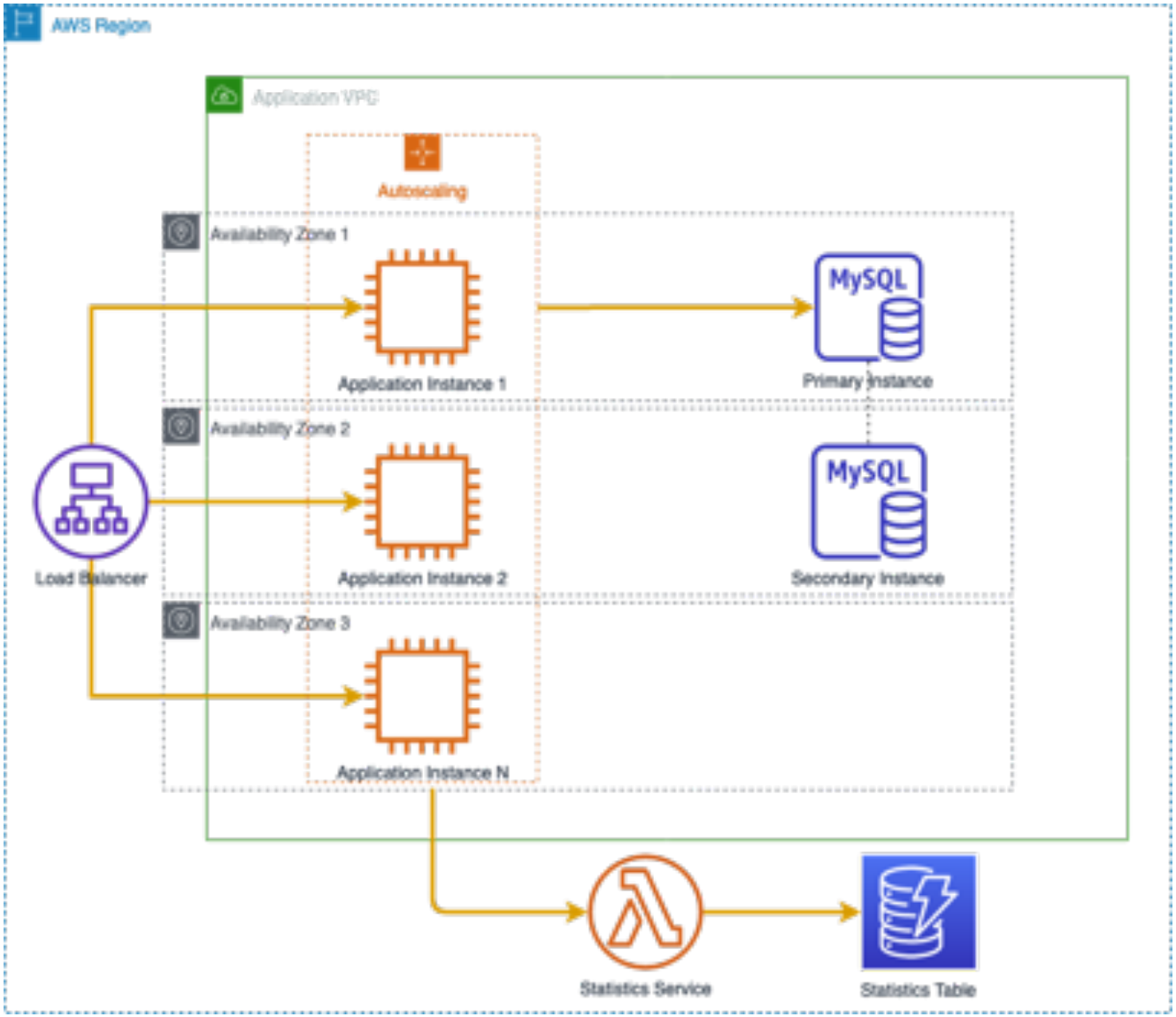


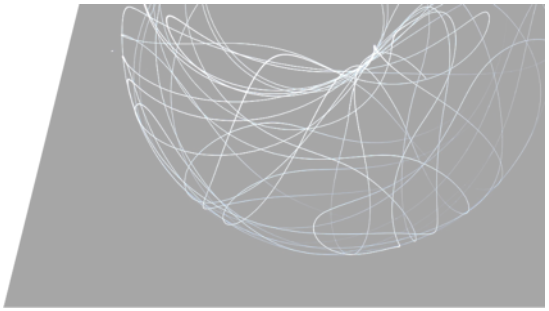
Failure Domain

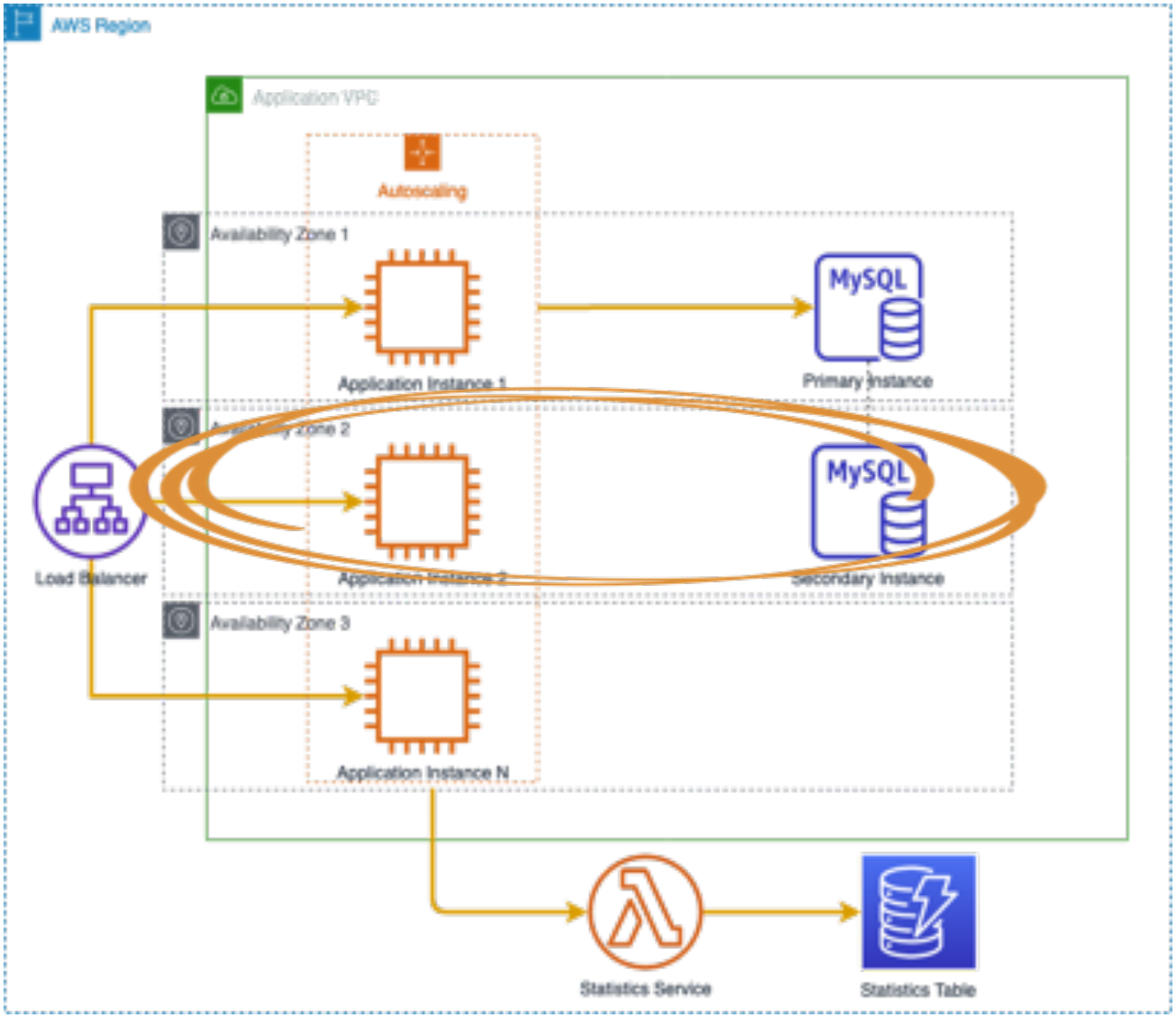


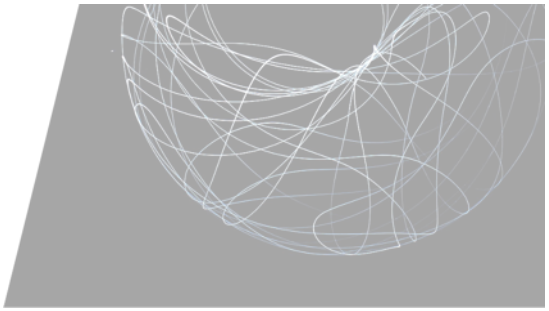
Scenario: Retrieve Web Page



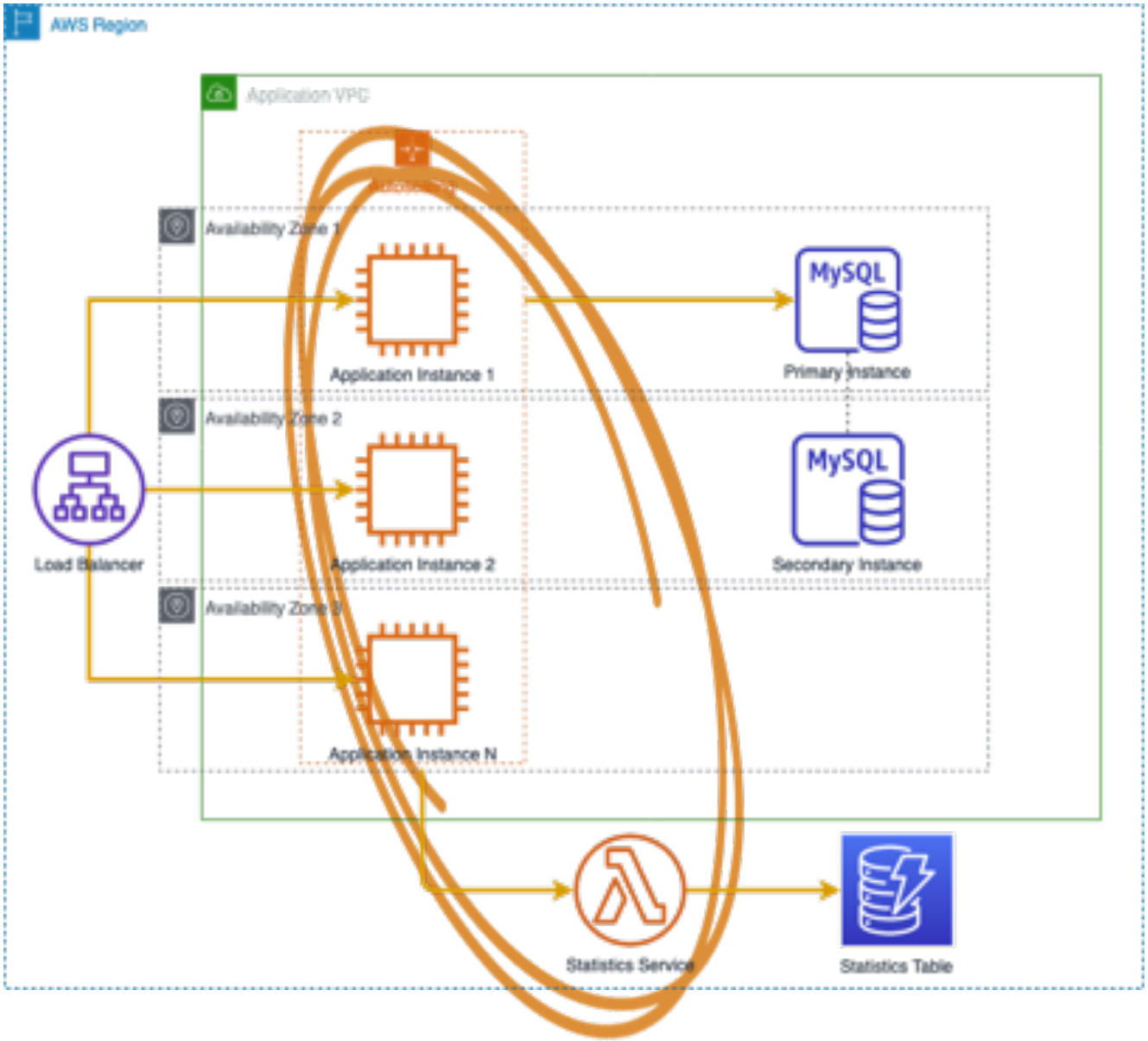

AWS Region
Application VPC
Autoscaling
Availability Zone 1
Application Instance 1
MySQL
Primary Instance
Availability Zone 2
Application Instance 2
MySQL
Secondary Instance
Load Balancer
Availability Zone 3
Application Instance N
Statistics Service
Statistics Table

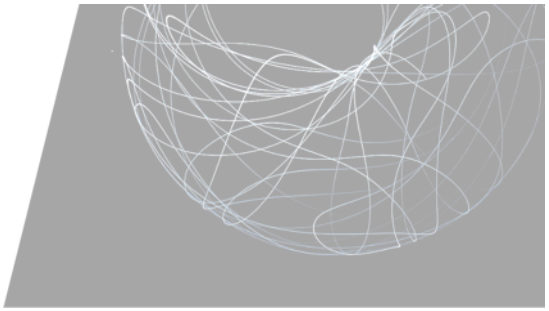






Failure domain of an AWS Availability Zone

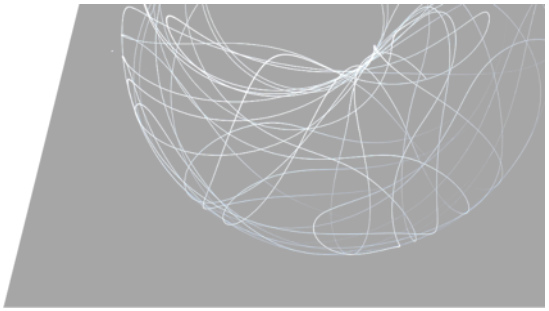




Failure domain of an AWS Service

Fault Isolation Boundaries

Failure-oriented Patterns

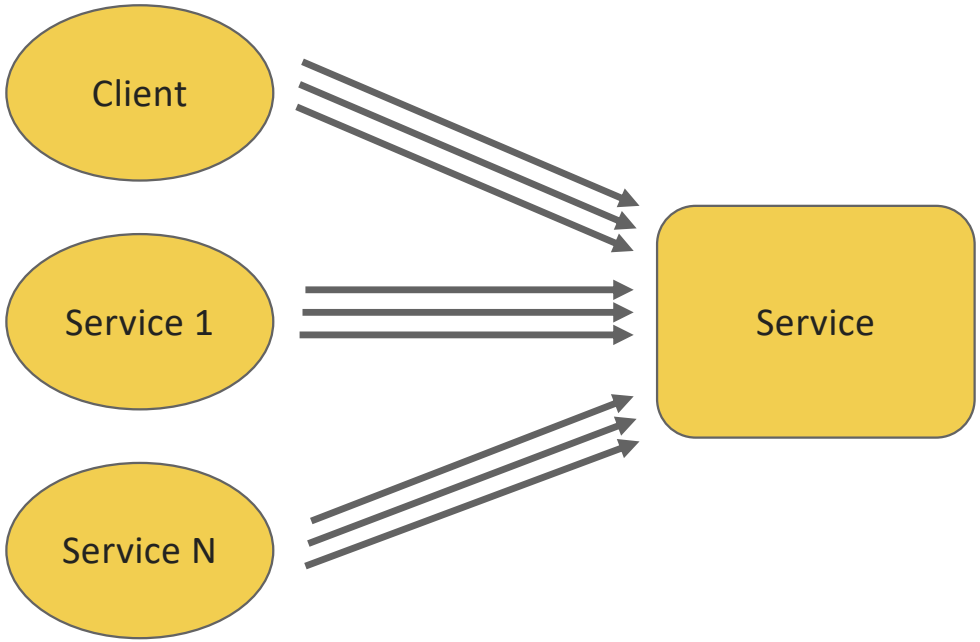


Client

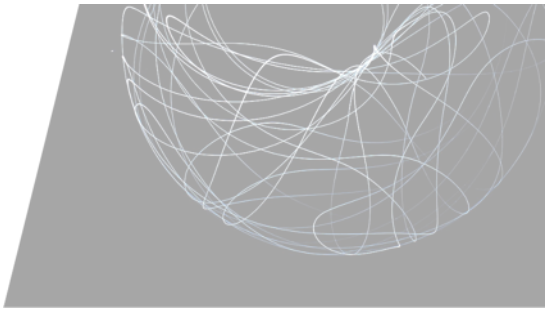
- Set timeouts
- Retries with backoff
  - Add jitter
  - Add retry limit
- Circuit breakers

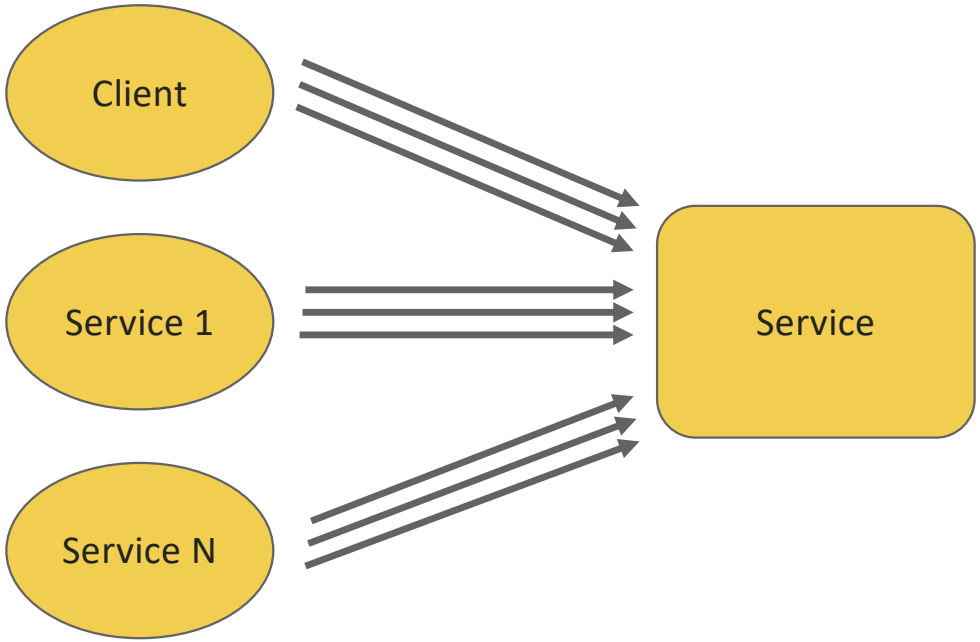
Service

- Rate limit
- Load shedding

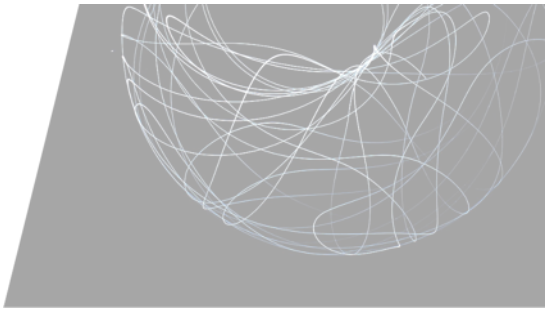


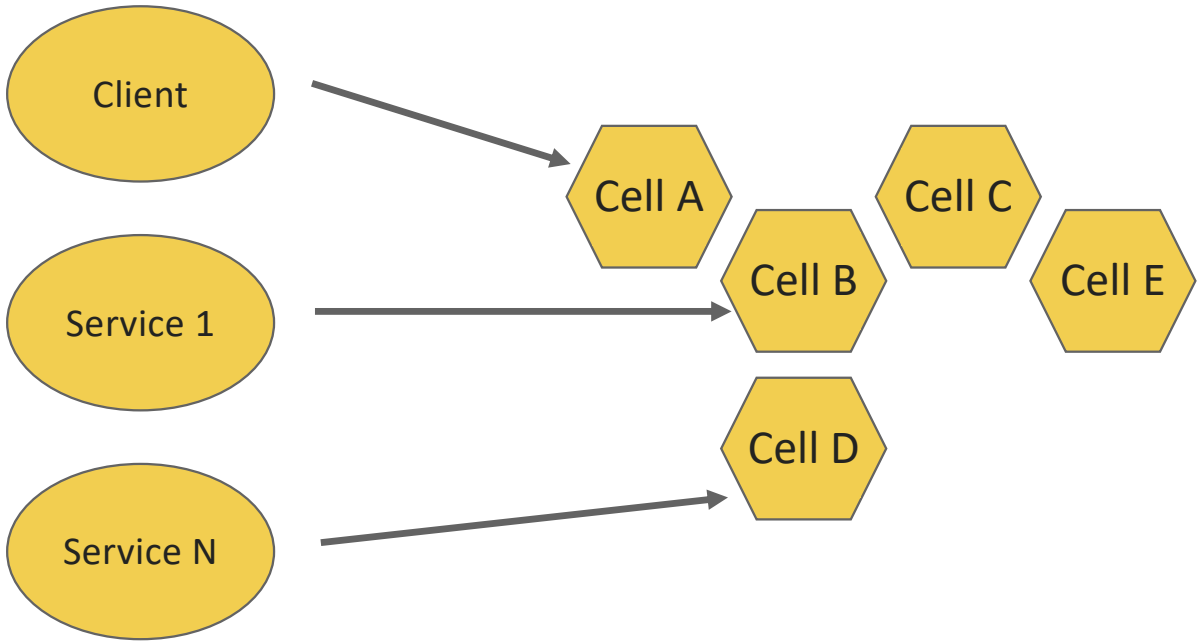
Failure-oriented Patterns



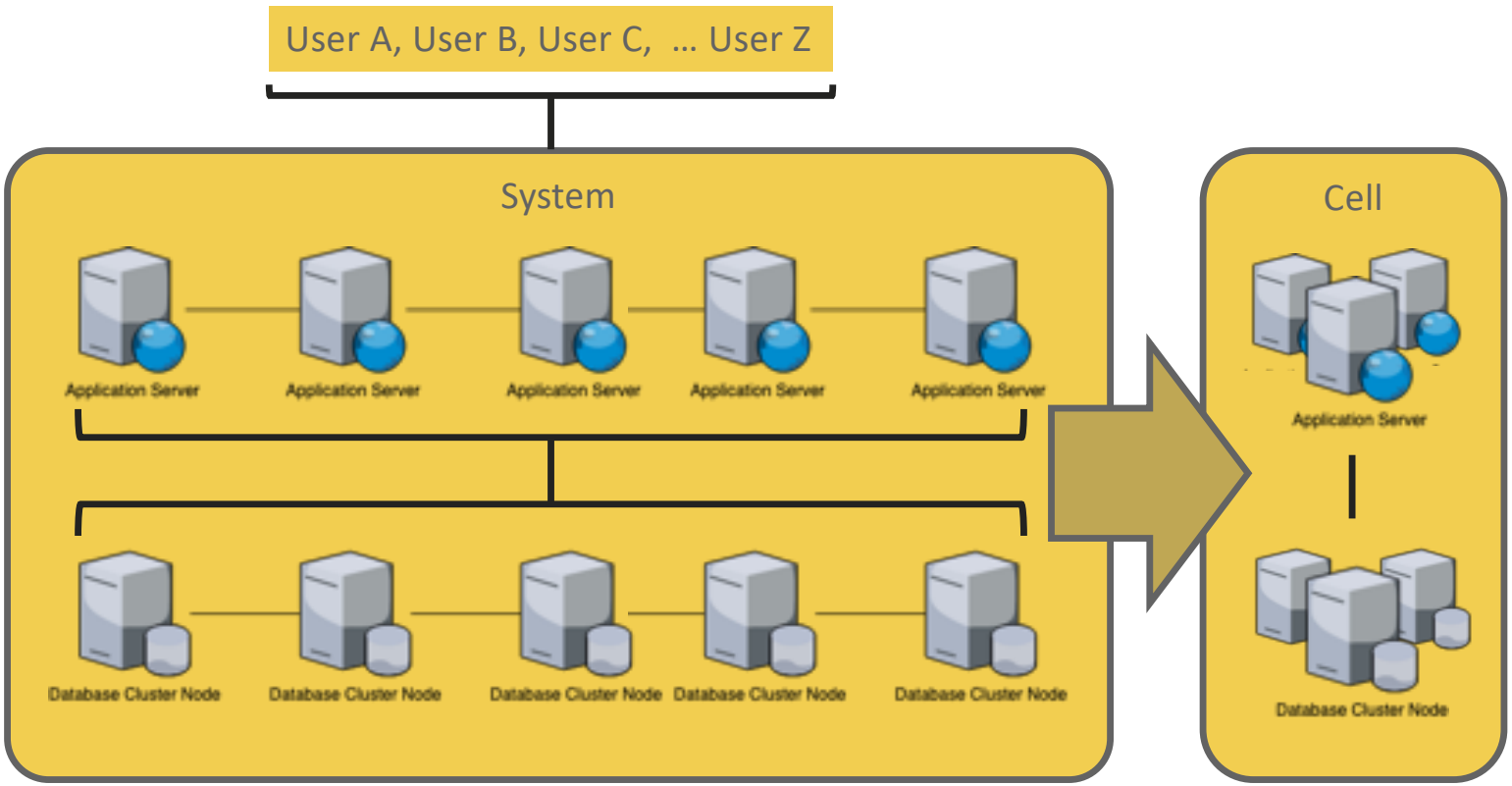


Limiting impact of failures with cells

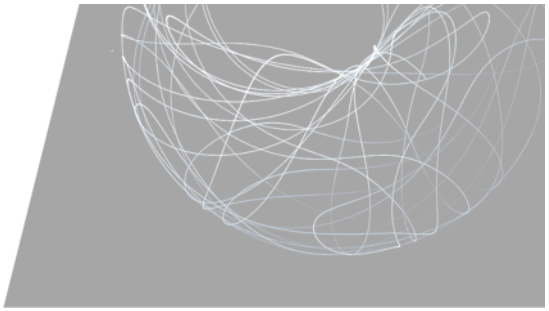


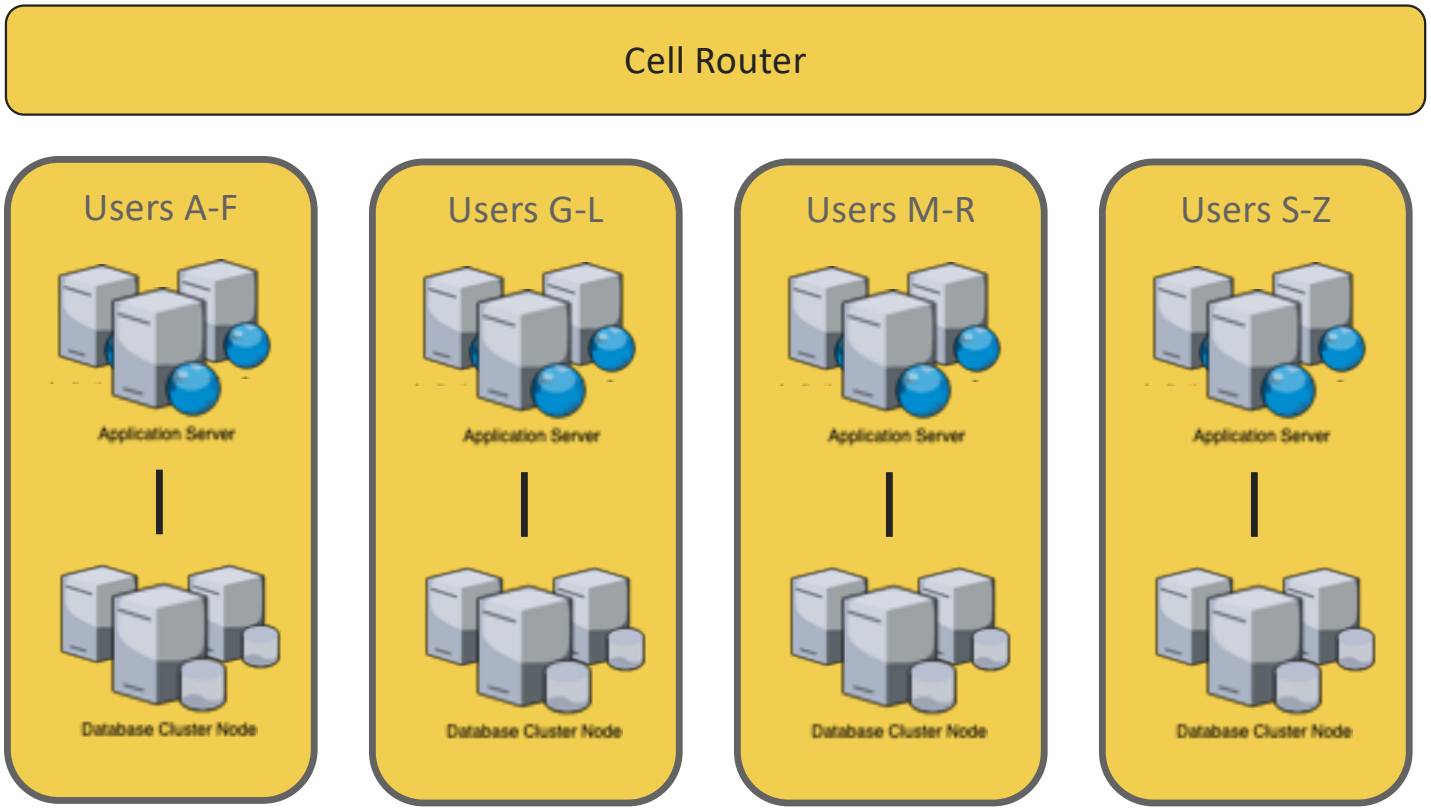


From System to Cell



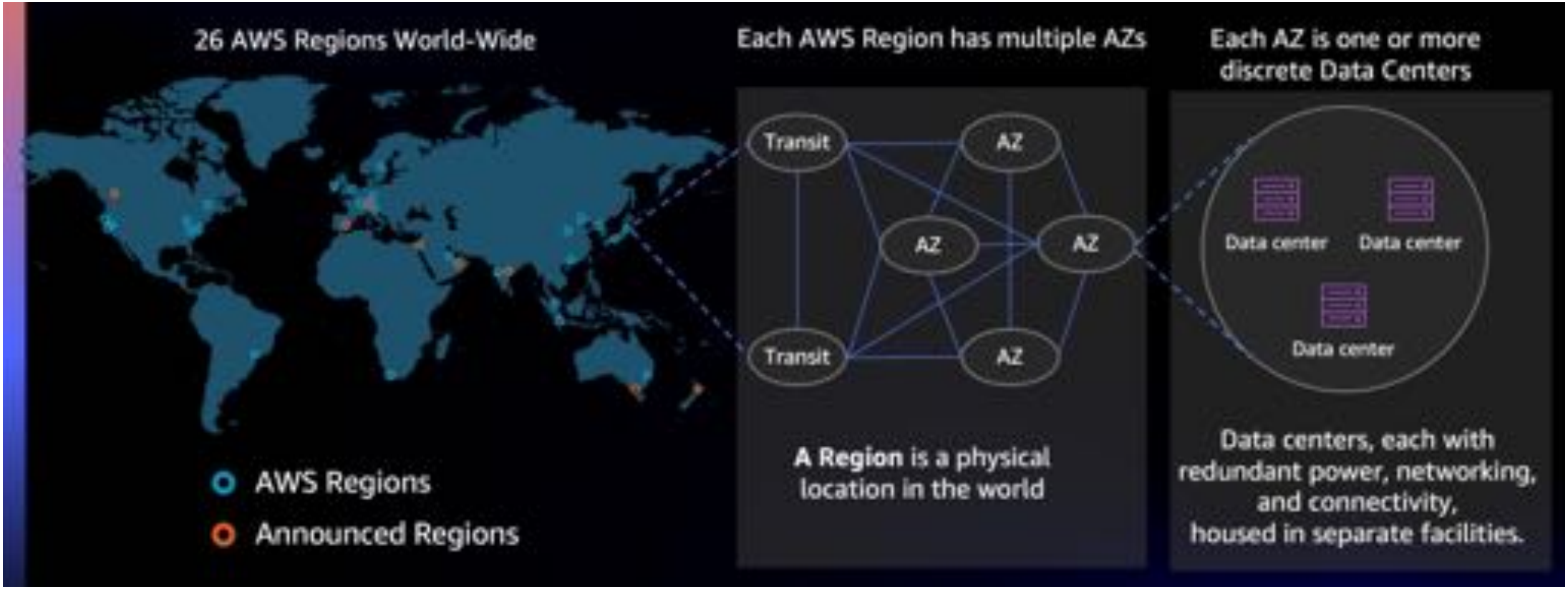
Cell-based Architecture

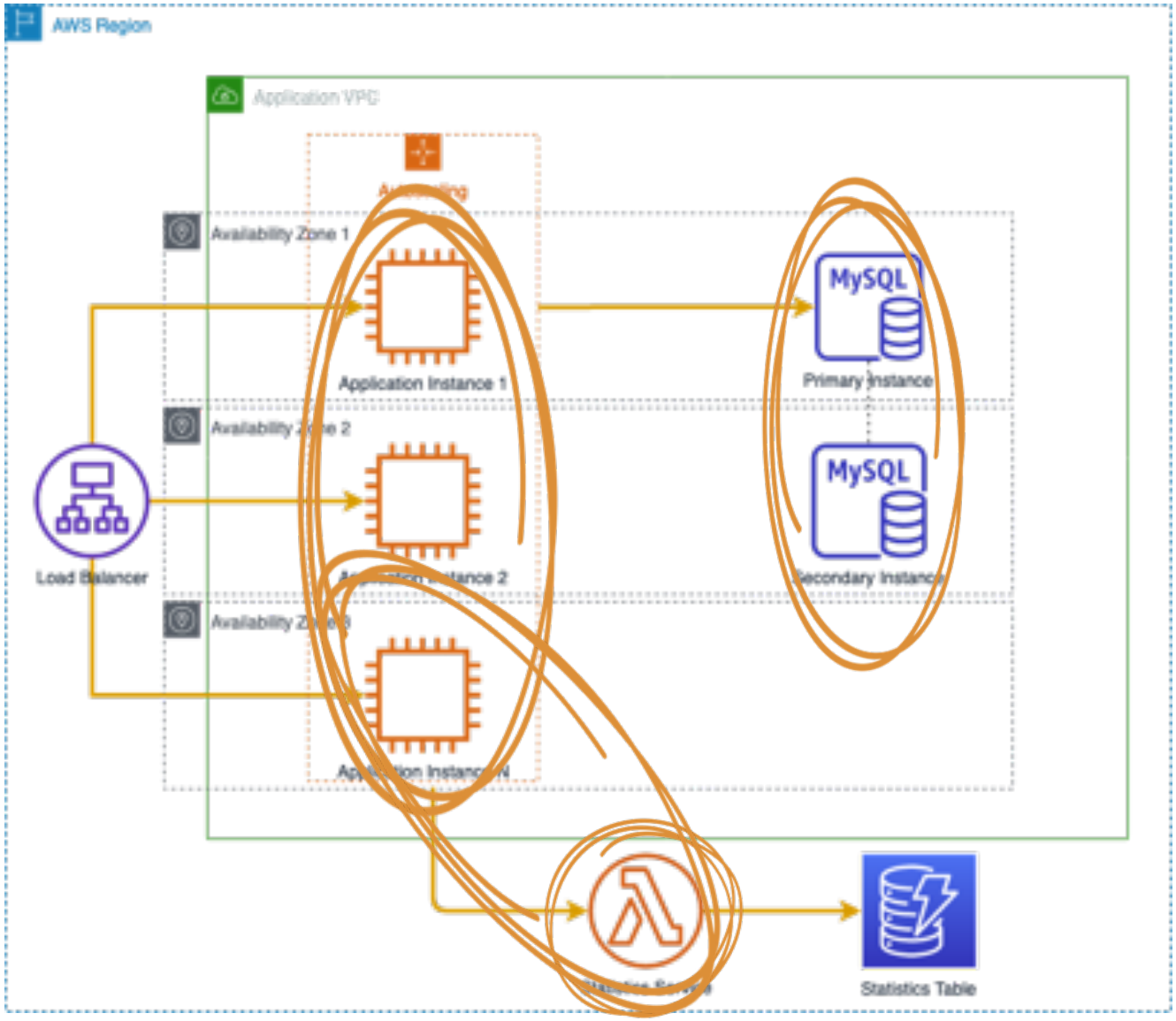


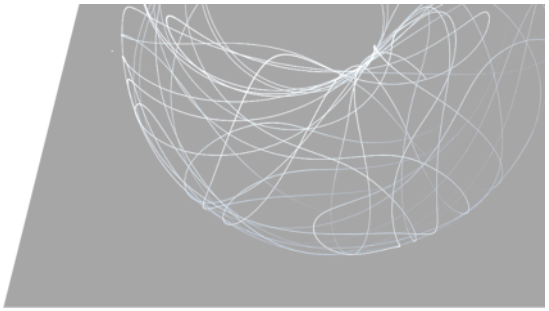


AWS Regions & Availability Zones (AZs)

AWS Regions are physical locations around the world with data center clusters



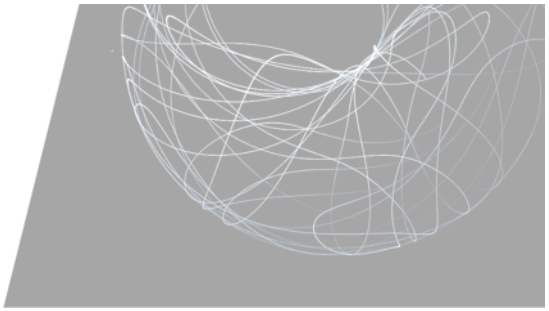




Fault isolation for our system

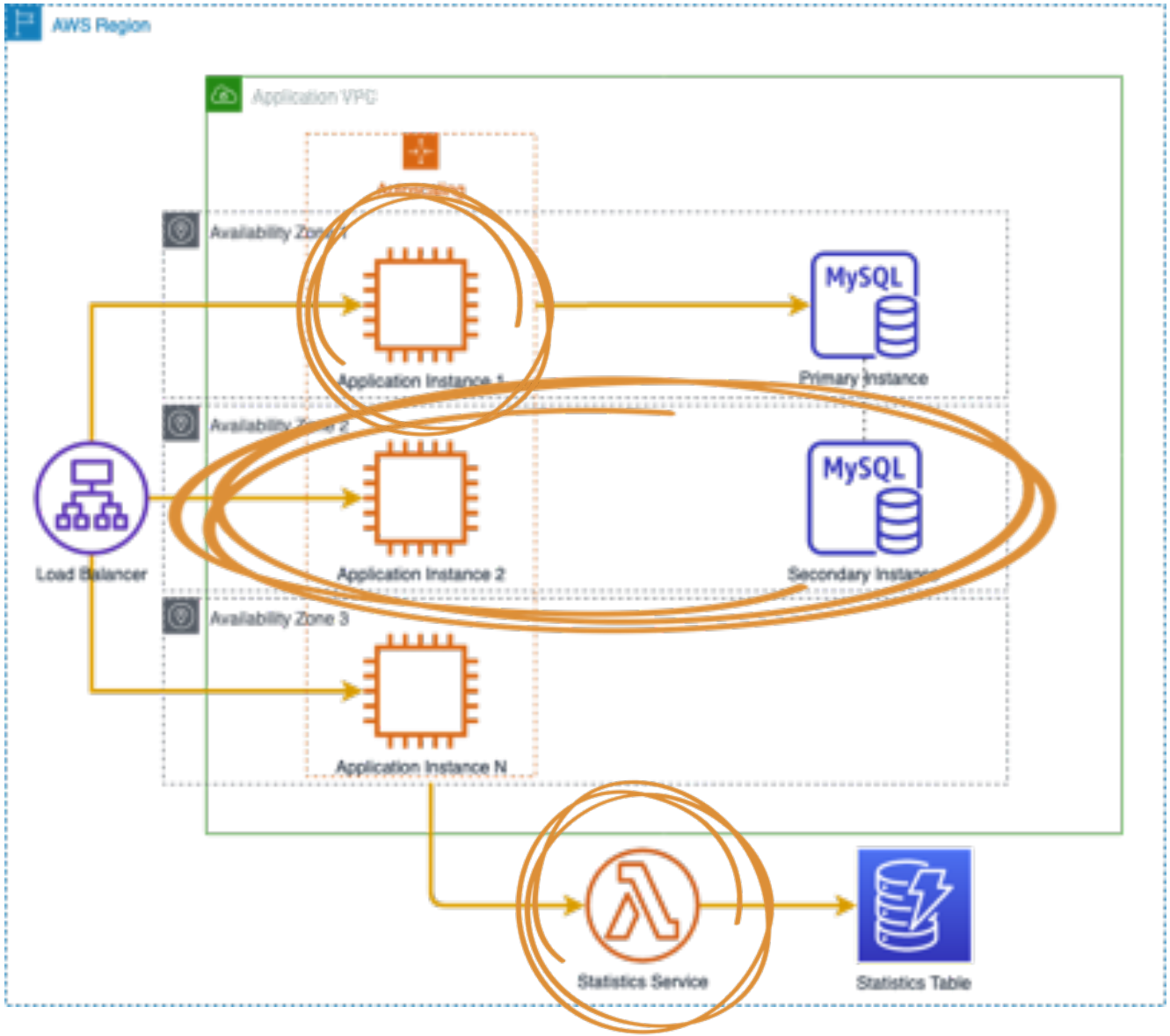
Initial Tests for Chaos Engineering

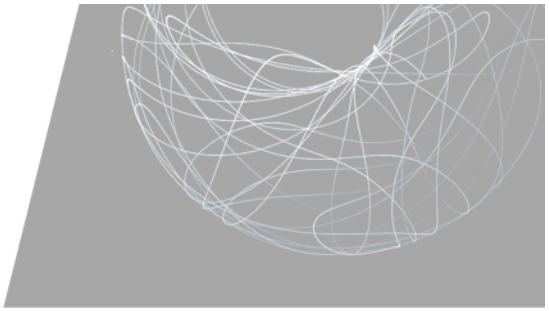
Disclaimer



Do not try this in Production

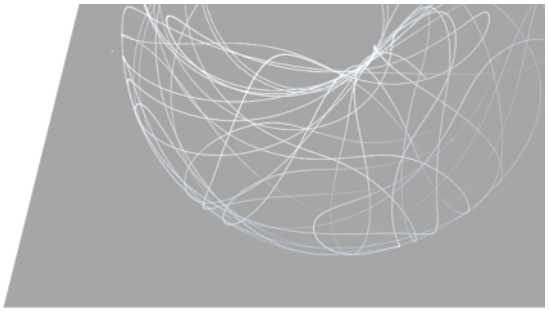
Initial Tests





- Zonal failure
- Centralized dependency
- Service dependency

Initial Tests: Zonal Resources



- Singular resource in an architecture (virtual machine, block device)

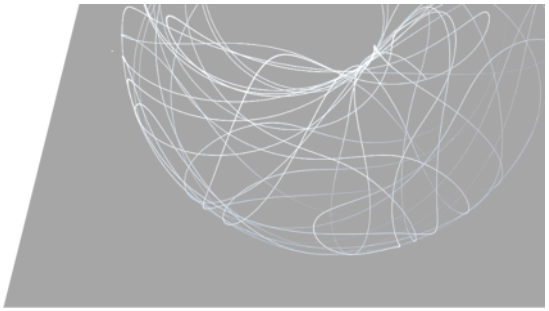
Terminate a virtual machine

```
$ aws ec2 terminate-instances --instance-ids i-1234567890abcdef0
```

Initiate a failover

```
$ aws rds reboot-db-instance \
      --db-instance-identifier my-db-identifier \
      --force-failover
```

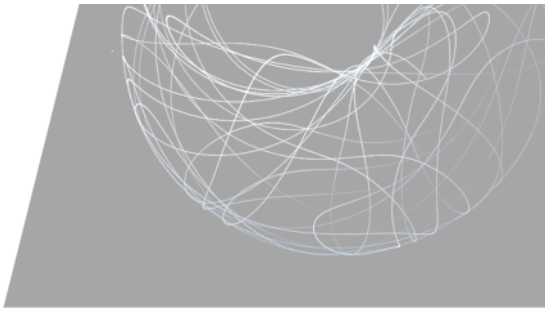
Initial Tests: Availability Zone



- Block all network traffic into / out of an availability zone

```
# Create a new NACL, storing the ID for the new NACL
$ NEW_NACL_ID=$(aws ec2 create-network-acl --vpc-id $VPC_ID \
  --query NetworkAcl.NetworkAclId --output text)

# Attach 2 rules to the new NACL that deny ingress and egress traffic
$ aws ec2 create-network-acl-entry --network-acl-id $NEW_NACL_ID \
  --rule-number 100 --cidr-block "0.0.0.0/0" --egress --protocol all \
  --port-range From=0,To=65535 --rule-action deny

$ aws ec2 create-network-acl-entry --network-acl-id $NEW_NACL_ID \
  --rule-number 101 --cidr-block "0.0.0.0/0" --ingress --protocol all \
  --port-range From=0,To=65535 --rule-action deny
```

Initial Tests: Regional Service



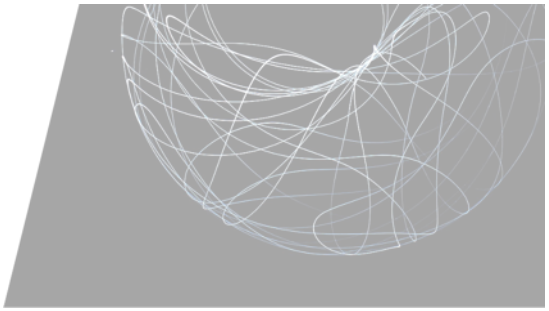
- Deny access to the service

```
# Define the policy document
$ cat >/tmp/deny-lambda.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        ”lambda:*"
      ],
      "Resource": "*"
    }
  ]
}
EOF
```

```
$ POLICY_ARN=$(aws iam \
  create-policy \
  --policy-name DenyLambda \
  --policy-document \
  file:///tmp/deny-lambda.json \
  --query 'Policy.Arn')
```

Demonstration

Chaos Engineering Tooling



AWS Fault Injection Simulator

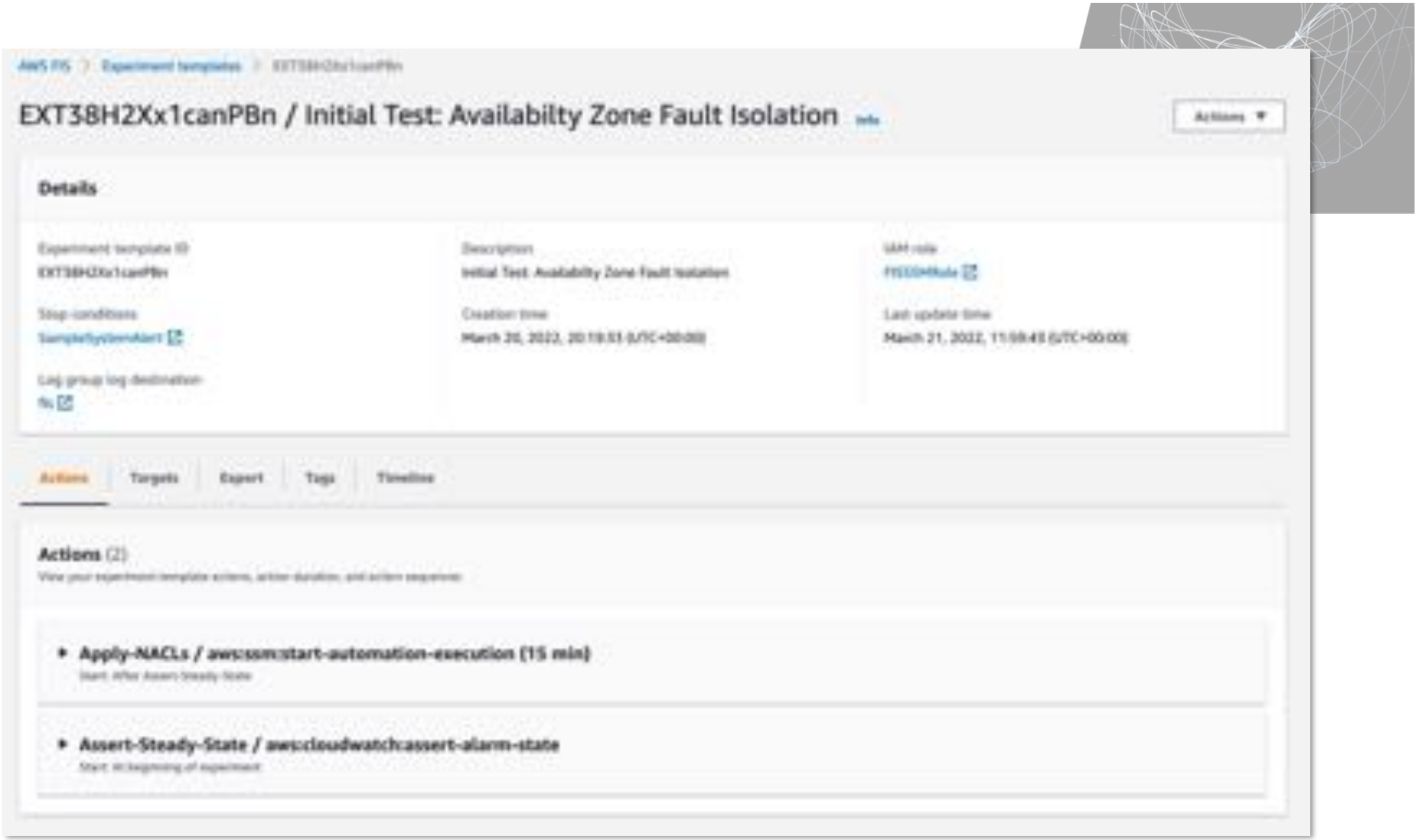
- Serverless
- Experiment templates
- Fault documents
- Integrated with Amazon CloudWatch

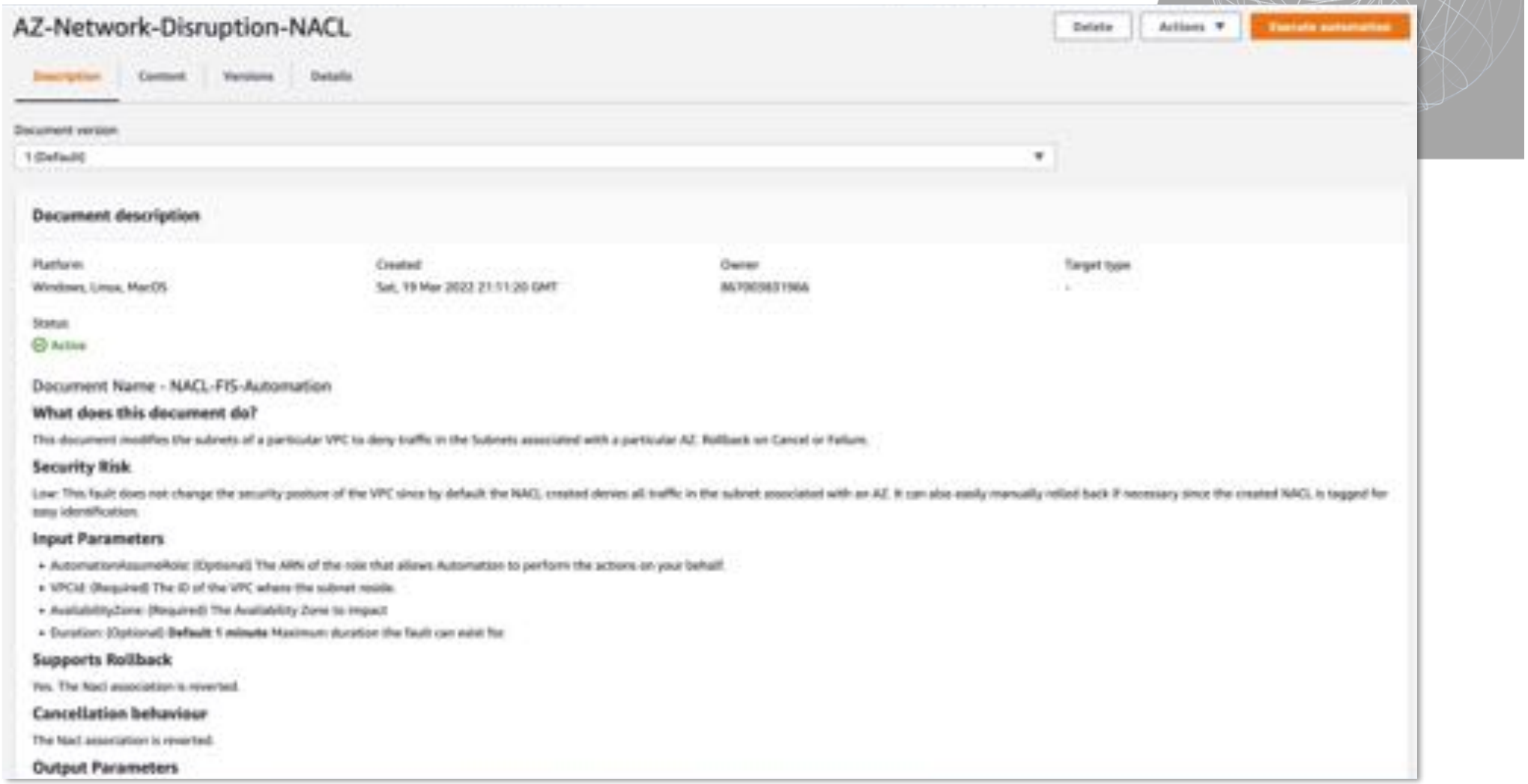
The Chaos Toolkit

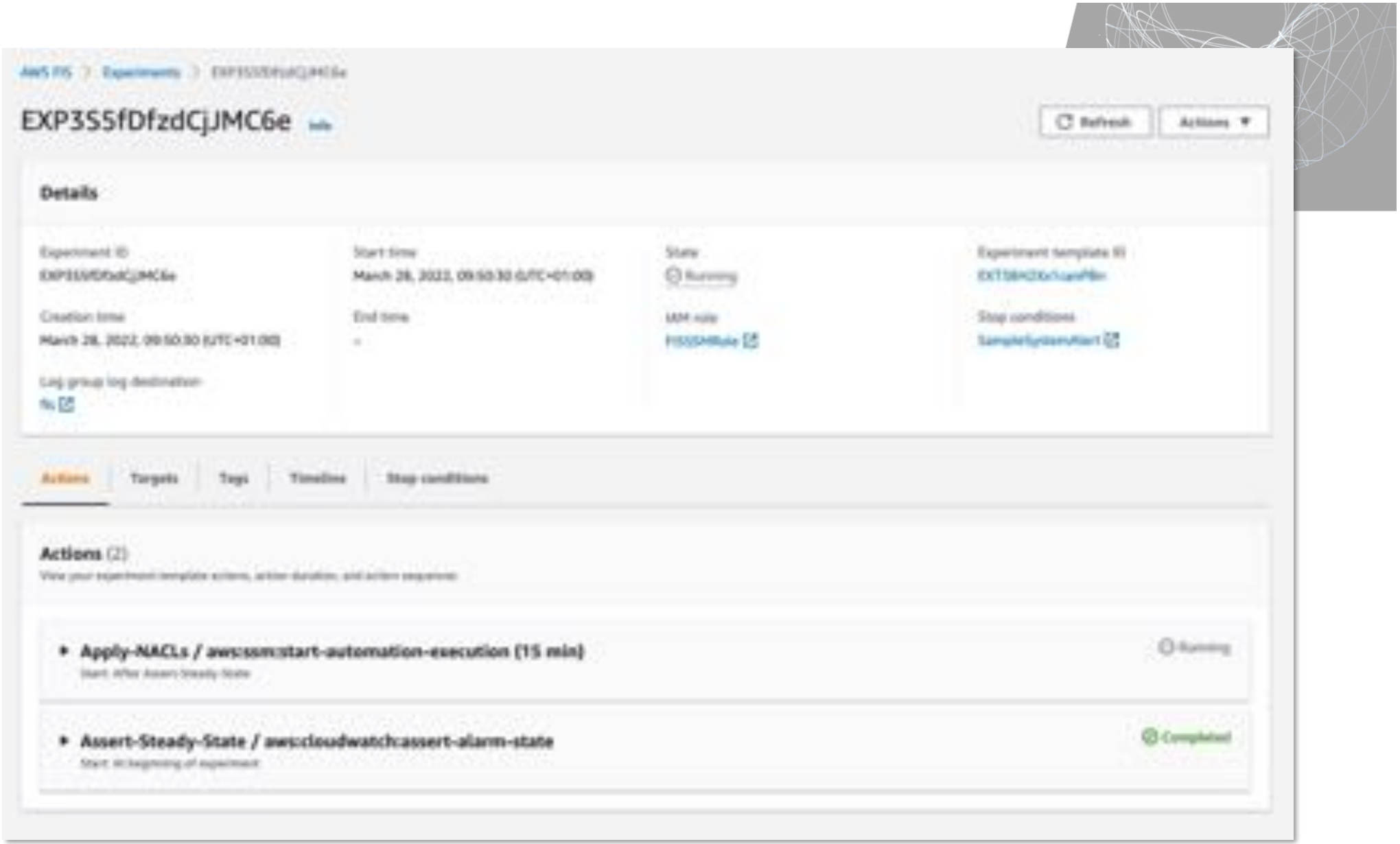
- An open API to chaos engineering
- Open source extensions for
  - Infrastructure/platform fault injections
  - Application fault injections
  - Observability
- Integrates easily into CI/CD pipelines

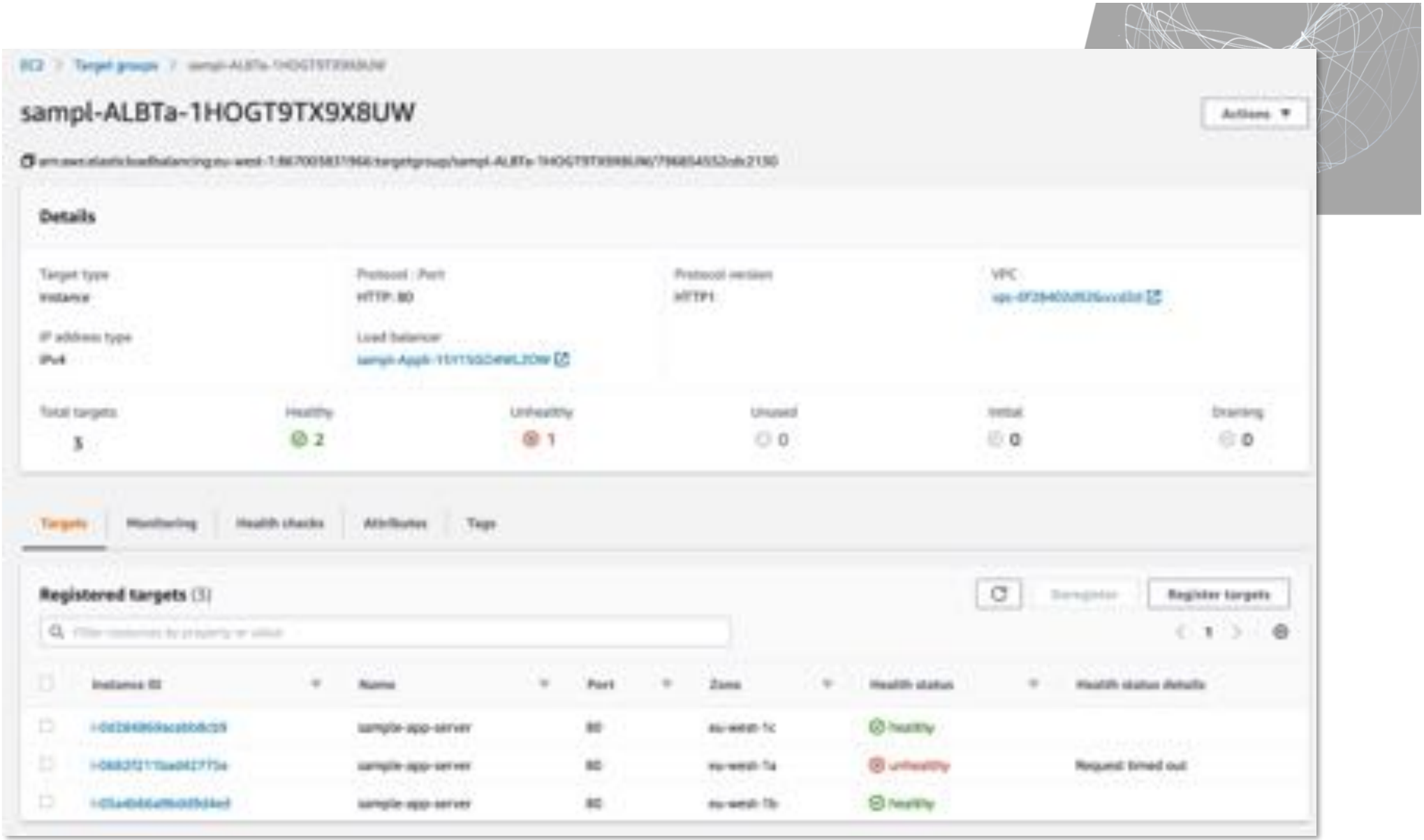

100
986
986

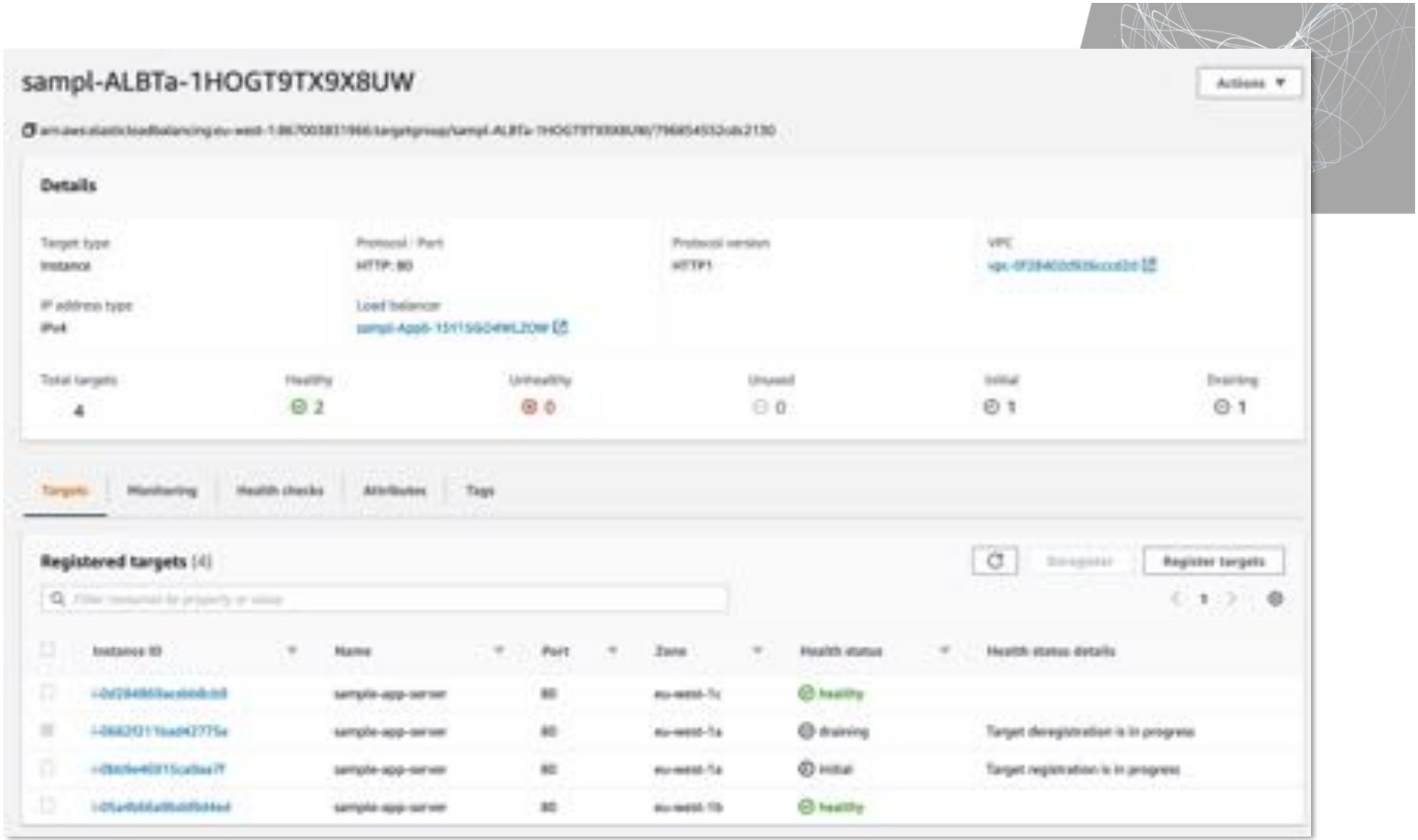
Simulate AZ Failure

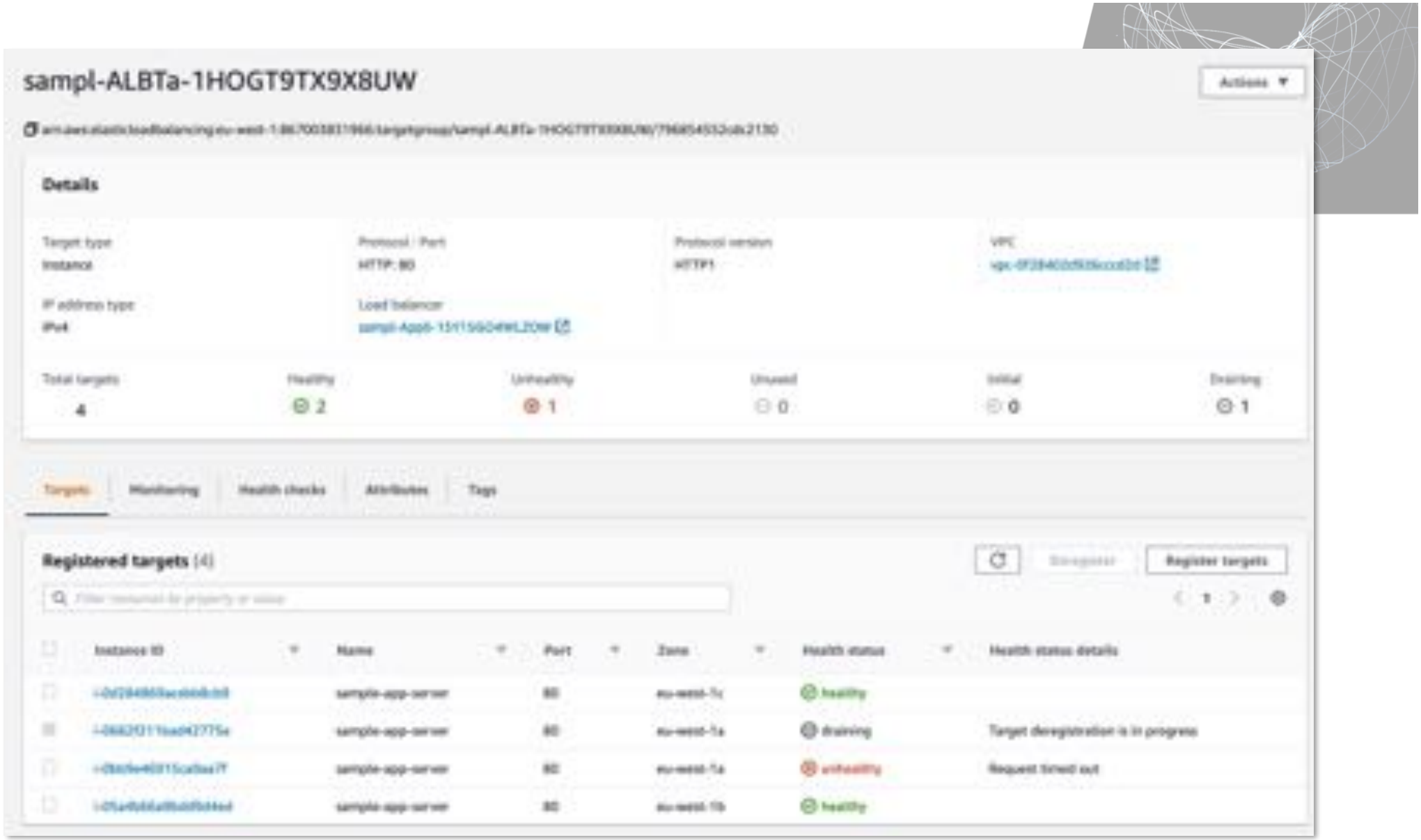

EXT38H2Xx1canPBn / Initial Test: Availabilty Zone Fault Isolation
Details
Actions
Targets
Export
Tags
Timeline
Actions (2)
Apply-NACLs / aws:ssm:start-automation-execution (15 min)
Assert-Steady-State / aws:cloudwatch:assert-alarm-state

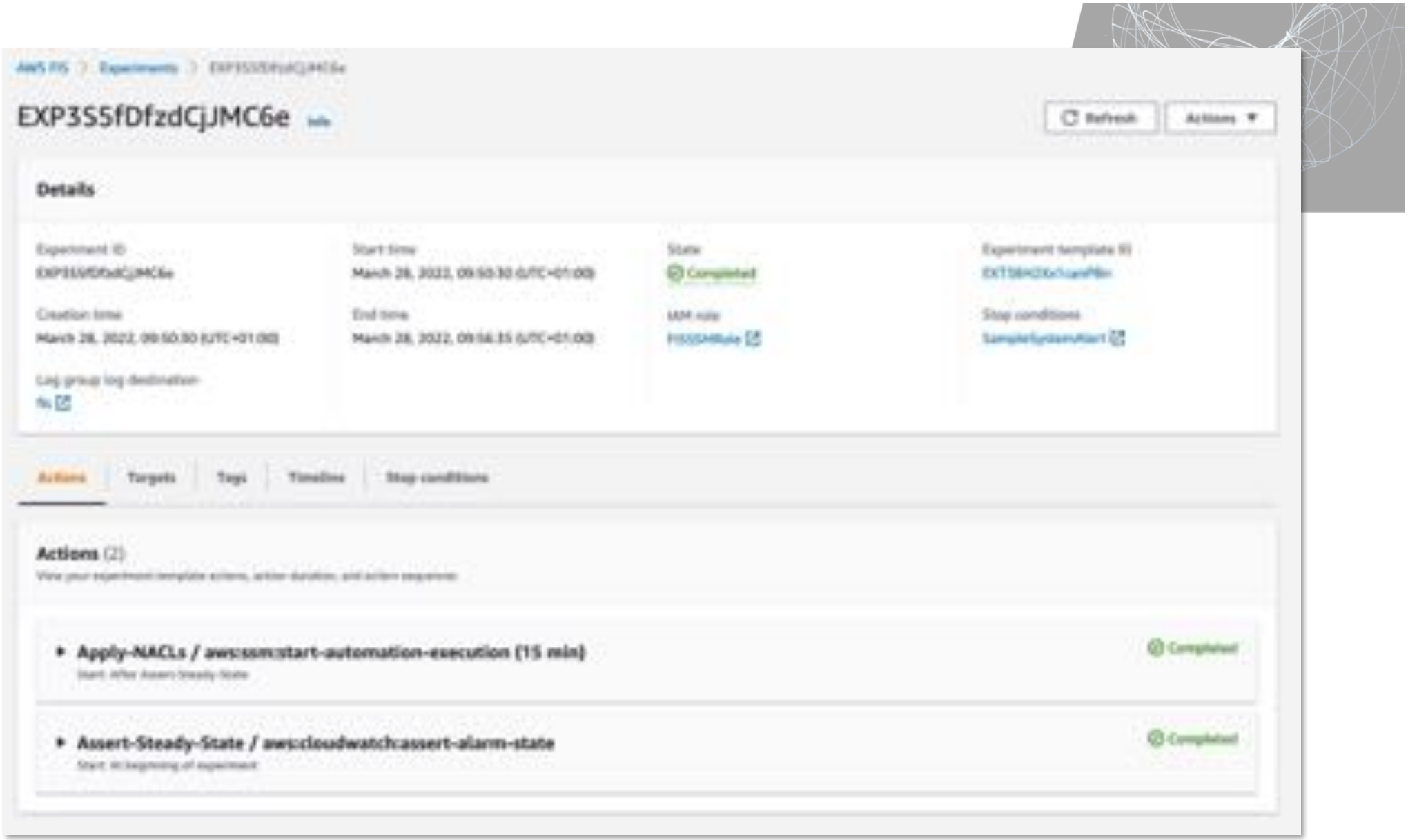

AZ-Network-Disruption-NACL
Delete
Actions
Document version
1 (Default)
Document description
Platform
Windows, Linux, MacOS
Created
Sat, 19 Mar 2022 21:11:20 GMT
Owner
Target type
Status
Active
Document Name - NACL-FIS-Automation
What does this document do?
This document modifies the subnets of a particular VPC to deny traffic in the Subnets associated with a particular AZ. Rollback on Cancel or Failure.
Security Risk
Low: This fault does not change the security posture of the VPC since by default the NACL created denies all traffic in the subnet associated with an AZ. It can also easily manually rolled back if necessary since the created NACL is tagged for easy identification.
Input Parameters
AutomationAssumeRole: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf.
VPCId: (Required) The ID of the VPC where the subnet reside.
AvailabilityZone: (Required) The Availability Zone to impact
Duration: (Optional) Default 1 minute Maximum duration the fault can exist for
Supports Rollback
Yes. The Nacl association is reverted.
Cancellation behaviour
The Nacl association is reverted.
Output Parameters


EXP3S5fDfzdCjJMC6e
Details
Actions
Targets
Tags
Timeline
Stop conditions
Actions (2)
Apply-NACLs / aws:ssm:start-automation-execution (15 min)
Assert-Steady-State / aws:cloudwatch:assert-alarm-state


sampl-ALBTa-1HOGT9TX9X8UW
Details
Target type
Instance
Protocol : Port
HTTP: 80
Protocol version
HTTP1
VPC
IP address type
IPv4
Load balancer
Total targets
3
Healthy
2
Unhealthy
1
Unused
0
Initial
0
Draining
0
Targets
Monitoring
Health checks
Attributes
Tags
Registered targets (3)
Register targets
Instance ID
Name
Port
Zone
Health status
Health status details
sample-app-server
80
us-west-1c
healthy
sample-app-server
80
us-west-1a
unhealthy
Request timed out
sample-app-server
80
us-west-1b
healthy


sampl-ALBTa-1HOGT9TX9X8UW
Actions
Details
Target type
Instance
Protocol : Port
HTTP: 80
Protocol version
HTTP1
VPC
IP address type
IPv4
Load balancer
Total targets
4
Healthy
2
Unhealthy
0
Unused
0
Initial
1
Draining
1
Targets
Monitoring
Health checks
Attributes
Tags
Registered targets (4)
Register targets
Instance ID
Name
Port
Zone
Health status
Health status details
sample-app-server
80
healthy
draining
Target deregistration is in progress
initial
Target registration is in progress

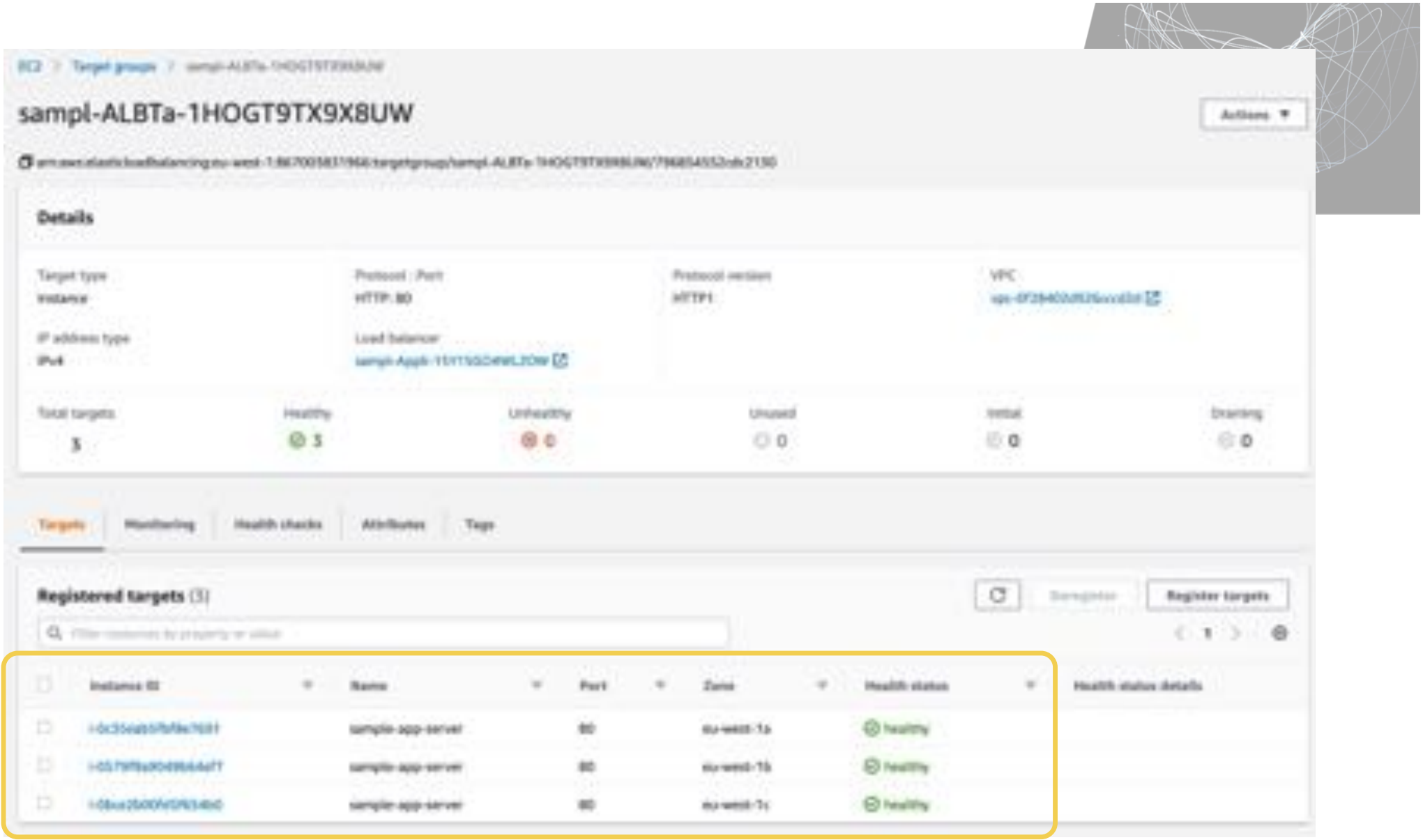

sampl-ALBTa-1HOGT9TX9X8UW
Actions
Details
Target type
Instance
Protocol : Port
HTTP: 80
Protocol version
HTTP1
VPC
IP address type
IPv4
Load balancer
Total targets
4
Healthy
2
Unhealthy
1
Unused
0
Initial
0
Draining
1
Targets
Monitoring
Health checks
Attributes
Tags
Registered targets (4)
Deregister
Register targets
Instance ID
Name
Port
Zone
Health status
Health status details
sample-app-server
80
healthy
sample-app-server
80
draining
Target deregistration is in progress
sample-app-server
80
unhealthy
Request timed out
sample-app-server
80
healthy

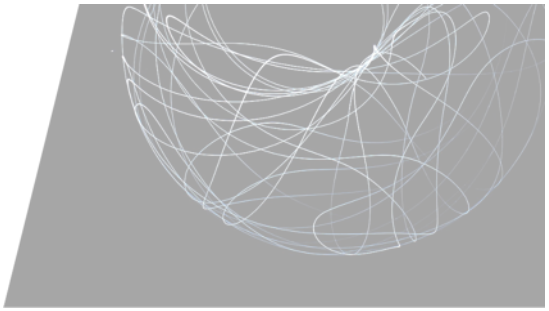

EXP3S5fDfzdCjJMC6e
Refresh
Actions
Details
Completed
Actions
Targets
Tags
Timeline
Stop conditions
Actions (2)
Apply-NACLs / aws:ssm:start-automation-execution (15 min)
Completed
Assert-Steady-State / aws:cloudwatch:assert-alarm-state
Completed


100
1.21 k
1.21 k

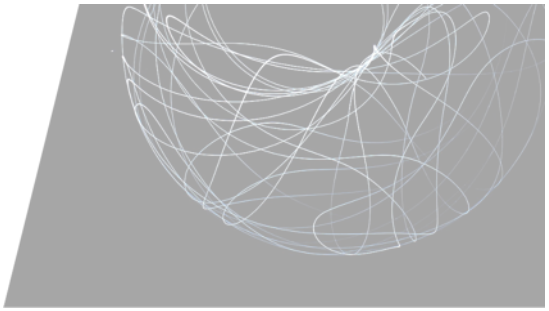
Simulate Region Failure


SampleSystemAlert
Pet Requests Served
100
HTTP 2XX Responses
1.09 k
Avg Response Time (sec)
HTTP Responses
1.09 k
Lambda Errors vs Invocations
DynamoDB Latency
Avg CPU Utilization
DB Avg CPU
Database Connections
3

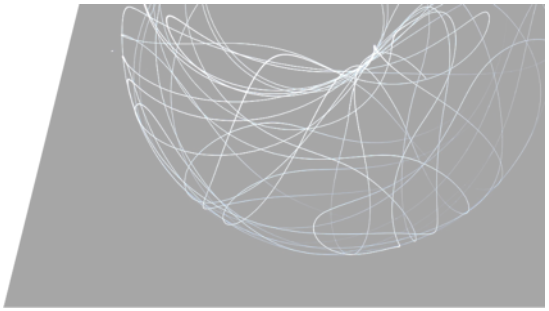

sampl-ALBTa-1HOGT9TX9X8UW
Actions
Details
Target type
Instance
Protocol : Port
HTTP: 80
Protocol version
HTTP1
VPC
IP address type
IPv4
Load balancer
Total targets
3
Healthy
3
Unhealthy
0
Unused
0
Initial
0
Draining
0
Targets
Monitoring
Health checks
Attributes
Tags
Registered targets (3)
Register targets
Instance ID
Name
Port
Zone
Health status
Health status details
sample-app-server
80
healthy



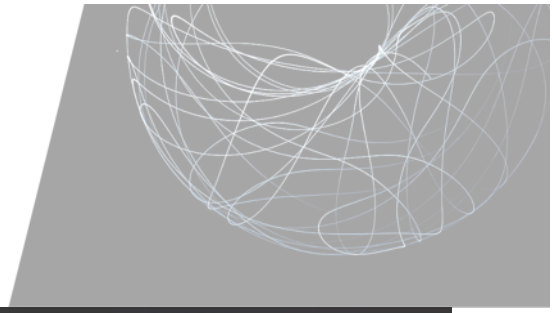
```
version: 1.0
title: Deny access to AWS Service
description: |
  To simulate service failure this experiment will apply an IAM policy with a DENY statement to restrict the service control plane
  to respond with a non-2xx response.

contributions:
  reliability: high
  availability: high

configuration:
  iam_policy_arn: arn:aws:iam::[ACCOUNT_ID]:policy/deny-my-eks-1-[ID]
  iam_role_name: "eksctl-my-eks-v1-NODEGROUP-[ID]"

steady-state-hypothesis:
  title: Service call success alarm is within tolerance
  probes:
    - type: probe
      name: check-system-error-rate
      tolerance: "OK"
      provider:
        type: python
        module: chaosaws.cloudwatch.probes
        func: get_alarm_state_value
        arguments:
          alarm_name: SampleSystemAlarm

method:
  - type: action
    name: attach-deny-policy
    provider:
      type: python
      module: chaosaws.iam.actions
      func: attach_role_policy
      arguments:
        arn: ${iam_policy_arn}
        role_name: ${iam_role_name}
    pauses:
      after: 900
  - ref: check-system-error-rate

rollbacks:
  - type: action
    name: detach-deny-policy
    provider:
      type: python
      module: chaosaws.iam.actions
      func: detach_role_policy
      arguments:
        arn: ${iam_policy_arn}
        role_name: ${iam_role_name}
```



```
[Finally-state-hypothesis]
  title: "Service call success alarm is within tolerance"
  probes:
    - type: probe
      name: check-system-error-rate
      tolerance: "OK"
      provider:
        type: python
        module: chaosaws.cloudwatch.probes
        func: get_alarm_state_value
        arguments:
          alarm_name: SampleAppLatAlarm
```



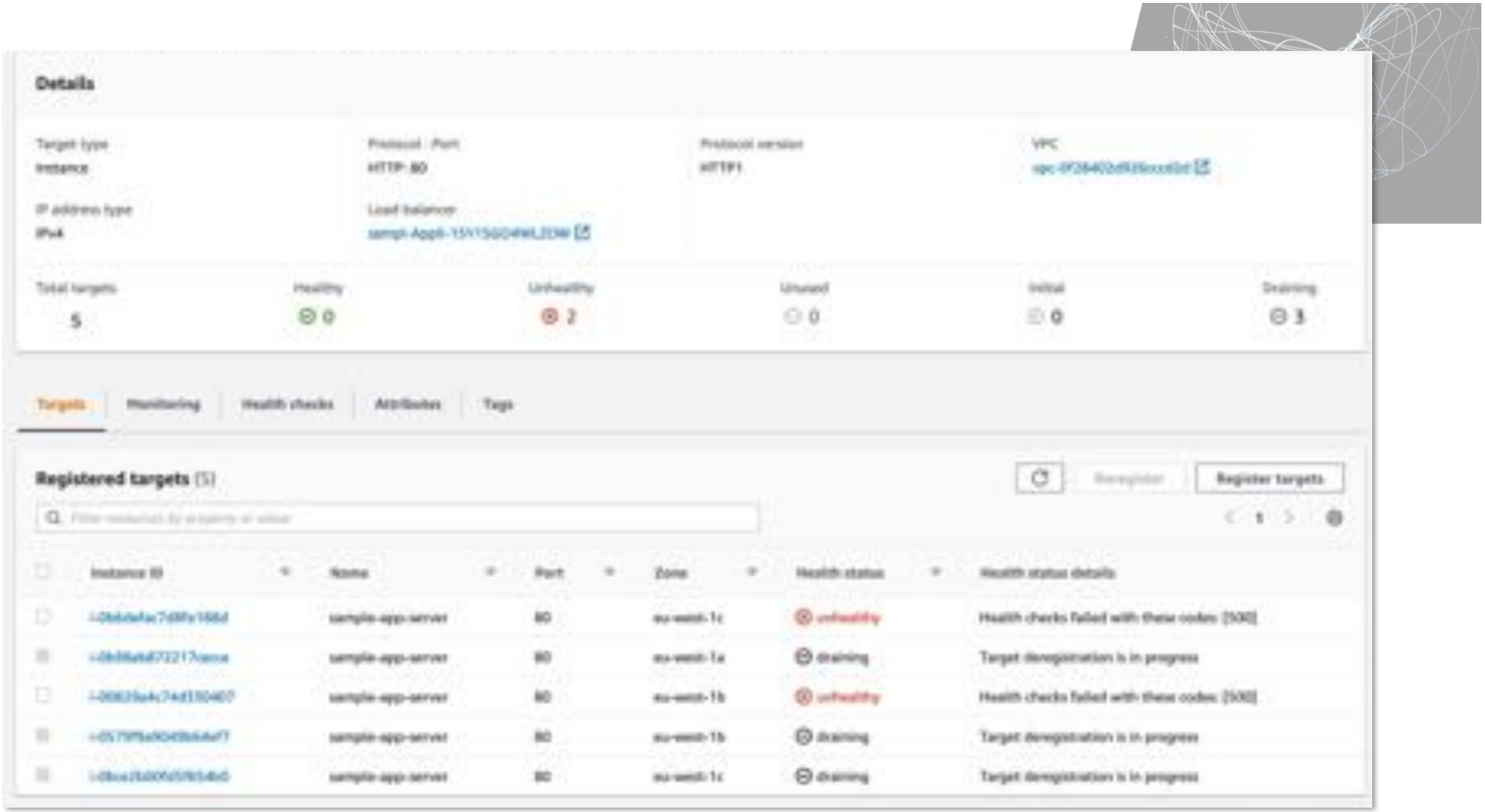
```
- type: action
  name: detach-deny-policy
  properties:
    type: python
    module: cloudcanvas.lambda_actions
    function: detach_role_policy
    arguments:
      - arn: $(iam_policy_arn)
      - role_name: $(iam_role_name)
```

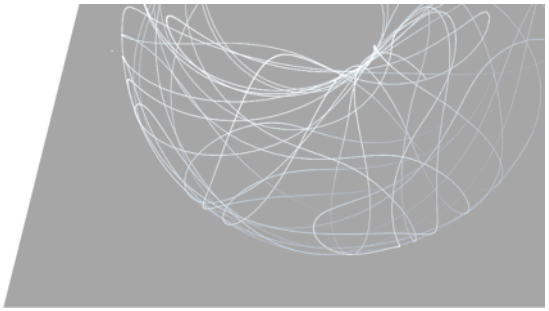


```
~/d/sample-aws-workload/e/fail-lambda on  main ?1 !2 ?9 chaos run experiment.yaml \
--rollback-strategy always \
--journal-path exp-fail-lambda-run-001.log
```

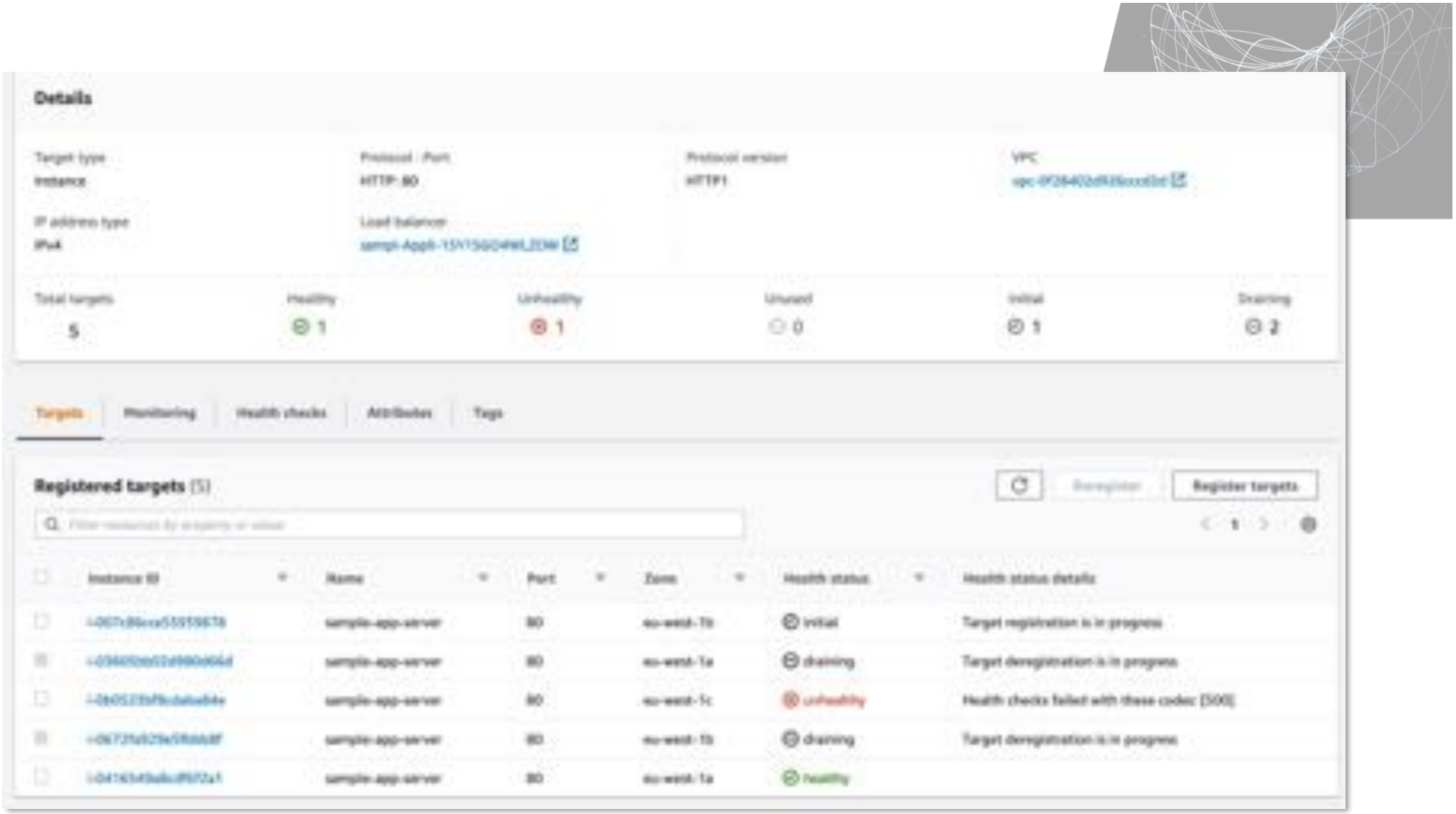
```
[2022-03-28 23:50:39 INFO] Validating the experiment's syntax
[2022-03-28 23:50:39 INFO] Experiment looks valid
[2022-03-28 23:50:39 INFO] Running experiment: Deny access to AWS Service
[2022-03-28 23:50:39 INFO] Steady-state strategy: default
[2022-03-28 23:50:39 INFO] Rollbacks strategy: always
[2022-03-28 23:50:39 INFO] Steady state hypothesis: Service call success alarm is within tolerance
[2022-03-28 23:50:39 INFO] Probe: check-system-error-rate
[2022-03-28 23:50:40 INFO] Steady state hypothesis is met!
[2022-03-28 23:50:40 INFO] Playing your experiment's method now...
[2022-03-28 23:50:40 INFO] Action: attach-deny-policy
[2022-03-28 23:50:40 INFO] Pausing after activity for 900s...
```


Pct Requests Served
7.84
HTTP 2XX Responses
95
Avg Response Time (sec)
HTTP Responses
1.21k
Lambda Errors vs Invocations
DynamoDB Latency
Avg CPU Utilization
DB Avg CPU
Database Connections
5

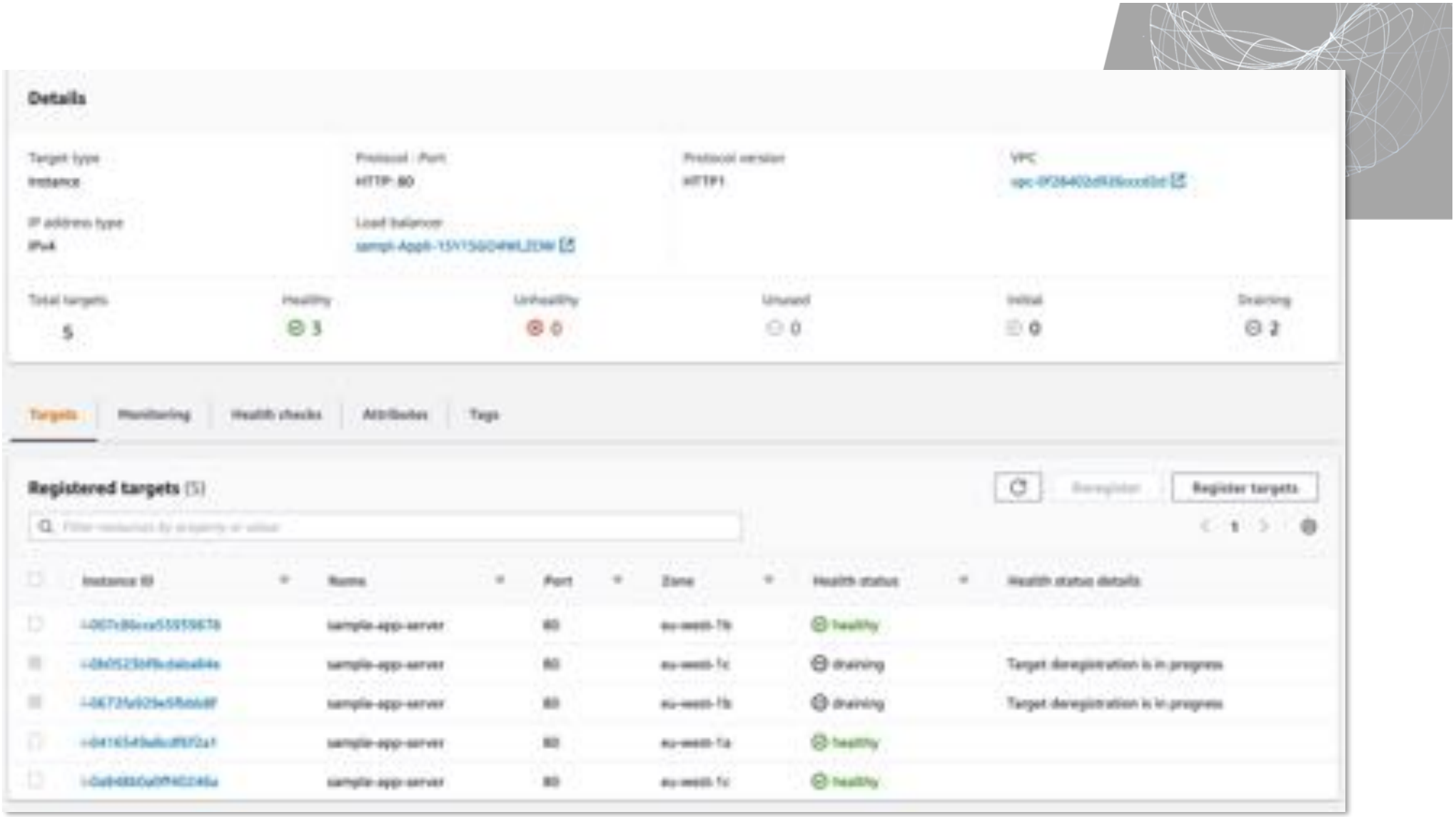

Details
Target type
Instance
Protocol : Port
HTTP: 80
Protocol version
HTTP1
VPC
IP address type
IPv4
Load balancer
Total targets
5
Healthy
0
Unhealthy
2
Unused
0
Initial
0
Draining
3
Targets
Monitoring
Health checks
Attributes
Tags
Registered targets (5)
Register targets
Instance ID
Name
Port
Zone
Health status
Health status details
sample-app-server
80
unhealthy
Health checks failed with these codes: [500]
sample-app-server
80
draining
Target deregistration is in progress
sample-app-server
80
unhealthy
Health checks failed with these codes: [500]
sample-app-server
80
draining
Target deregistration is in progress
sample-app-server
80
draining
Target deregistration is in progress



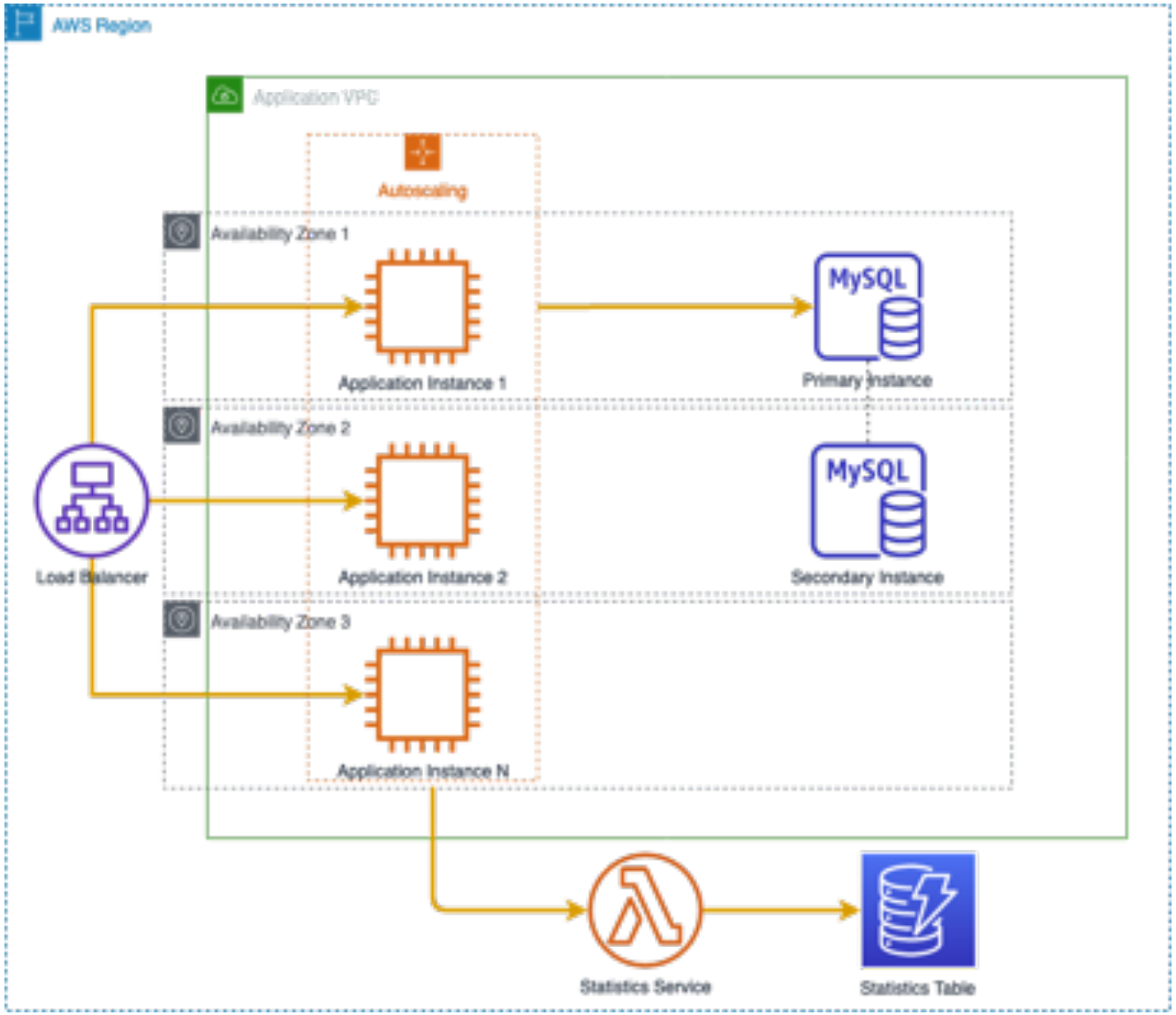
```
[2022-03-28 23:50:39 INFO] Validating the experiment's syntax
[2022-03-28 23:50:39 INFO] Experiment looks valid
[2022-03-28 23:50:39 INFO] Running experiment: Deny access to AWS Service
[2022-03-28 23:50:39 INFO] Steady-state strategy: default
[2022-03-28 23:50:39 INFO] Rollbacks strategy: always
[2022-03-28 23:50:39 INFO] Steady state hypothesis: Service call success alarm is within tolerance
[2022-03-28 23:50:39 INFO] Probe: check-system-error-rate
[2022-03-28 23:50:40 INFO] Steady state hypothesis is met!
[2022-03-28 23:50:40 INFO] Playing your experiment's method now...
[2022-03-28 23:50:40 INFO] Action: attach-deny-policy
[2022-03-28 23:50:40 INFO] Pausing after activity for 900s...
[2022-03-29 00:06:07 INFO] Probe: check-system-error-rate
[2022-03-29 00:06:07 INFO] Steady state hypothesis: Service call success alarm is within tolerance
[2022-03-29 00:06:07 INFO] Probe: check-system-error-rate
[2022-03-29 00:06:08 CRITICAL] Steady state probe 'check-system-error-rate' is not in the given tolerance so failing this experiment
[2022-03-29 00:06:08 WARNING] Rollbacks were explicitly requested to be played
[2022-03-29 00:06:08 INFO] Let's rollback...
[2022-03-29 00:06:08 INFO] Rollback: detach-deny-policy
[2022-03-29 00:06:08 INFO] Action: detach-deny-policy
[2022-03-29 00:06:08 INFO] Experiment ended with status: deviated
[2022-03-29 00:06:08 INFO] The steady-state has deviated, a weakness may have been discovered
```

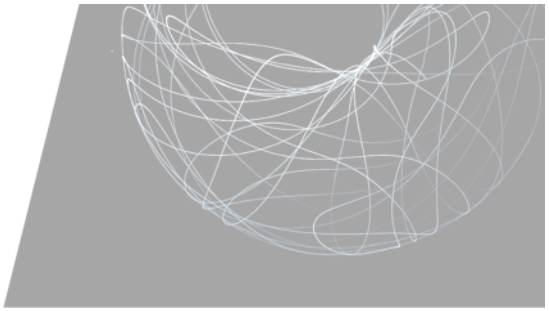

Details
Target type
Instance
Protocol : Port
HTTP: 80
Protocol version
HTTP1
VPC
IP address type
IPv4
Load balancer
Total targets
5
Healthy
1
Unhealthy
1
Unused
0
Initial
1
Draining
2
Targets
Monitoring
Health checks
Attributes
Tags
Registered targets (5)
Register targets
Instance ID
Name
Port
Zone
Health status
Health status details
sample-app-server
80
Initial
Target registration is in progress
sample-app-server
80
draining
Target deregistration is in progress
sample-app-server
80
unhealthy
Health checks failed with these codes: [500]
sample-app-server
80
draining
Target deregistration is in progress
sample-app-server
80
healthy


SampleSystemAlert
Pct Requests Served
100
HTTP 2XX Responses
601
Avg Response Time (sec)
HTTP Responses
601
Lambda Errors vs Invocations
DynamoDB Latency
Avg CPU Utilization
DB Avg CPU
Database Connections
5


Details
Target type
Instance
Protocol : Port
HTTP: 80
Protocol version
HTTP1
VPC
IP address type
IPv4
Load balancer
Total targets
5
Healthy
3
Unhealthy
0
Unused
0
Initial
0
Draining
2
Targets
Monitoring
Health checks
Attributes
Tags
Registered targets (5)
Register targets
Instance ID
Name
Port
Zone
Health status
Health status details
sample-app-server
80
healthy
draining
Target deregistration is in progress
sample-app-server
80
draining
Target deregistration is in progress
sample-app-server
80
healthy
sample-app-server
80
healthy
sample-app-server
80
healthy

Wrapping Up

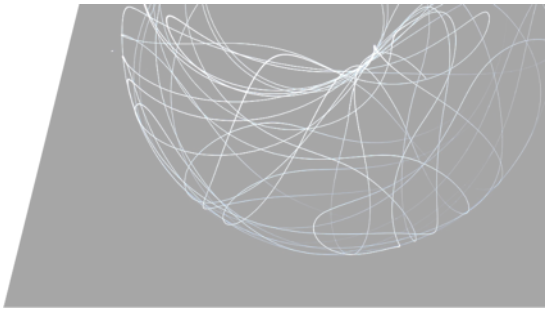




Learnings:

- Centralized dependency
- Service dependency

Wrap Up



- Nature of failure (binary vs gray failure)
- This is only an initial set of tests
- Identify your failure domains and fault isolation boundaries
- Test your fault isolation boundaries

jason_barto